Solution 2: Publish a SOAP web service

1. **Create a web service package.**

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

    b) Enter **webservice.company** as the new package name.

2. **Create EmployeeSummary class return type.**

    a) Right-click on the **company** package and select **New ➔ Gosu Class**.

    b) Enter **EmployeeSummary** as the new Gosu class name.

```
package si.ta.webservice.company

uses gw.xml.ws.annotation.WsiExportable

/**
 * Created by training.
 */
@WsiExportable
final class EmployeeSummary {
  // class properties
  var _numberOfEmployees: int as NumberOfEmployees
  var _employeeScore: int as EmployeeScore
  var _headquartersLocation: String as HeadquartersLocation
}
```

3. **Create CompanyAPI Gosu class.**

    a) Right-click on the **company** package and select **New ➔ Gosu Class**.

    b) Enter **CompanyAPI** as the new Gosu class name.

4. **Create CompanyAPI methods.**

```
package si.ta.webservice.company

uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.transaction.Transaction
uses gw.xml.ws.annotation.WsiWebService

/**
 * Created by training.
 */
@WsiWebService
class CompanyAPI {

  /**
   * Function verifies if a company exists.
   */
  function doesCompanyExist(taxID: String): boolean {
    // query for Company for a given taxID
    var targetCompany = findCompanyByTaxID(taxID)
    if (targetCompany != null) {
      return true
    } else {
      return false
    }
  }

  /**
```

```
   * Function creates a ContactNote for a given company.
   */
  @Param("taxID", "Company taxID")
  @Param("body", "String identifying the body of the note")
  function createContactNote(taxID: String, body: String): void {
    // query for Company for a given taxID
    var targetCompany = findCompanyByTaxID(taxID)
    if (targetCompany != null){
      // create new bundle
      Transaction.runWithNewBundle(\newBundle -> {
        // add query read-only object to newBundle
        targetCompany = newBundle.add(targetCompany)
        // create new Note and add to Company
        var newNote = new ContactNote()
        newNote.ContactNoteType = typekey.ContactNoteType.TC_GENERAL
        newNote.Subject = "External Note"
        newNote.Body = body
        targetCompany.addToContactNotes(newNote)
      })
    }
  }

  /**
   * Function returns an EmployeeSummary object for a given tax ID.
   */
  @Param("taxID", "Company tax ID")
  @Returns("EmployeeSummary object")
  function getEmployeeSummary(taxID: String): EmployeeSummary {
    // query for Company for a given taxID
    var targetCompany = findCompanyByTaxID(taxID)
    if (targetCompany != null) {
      var anEmployeeSummary = new EmployeeSummary()
      anEmployeeSummary.EmployeeScore = targetCompany.EmployeeScore
      anEmployeeSummary.NumberOfEmployees = targetCompany.NumberOfEmployees
      anEmployeeSummary.HeadquartersLocation = targetCompany.PrimaryAddress.City
          + "," + targetCompany.PrimaryAddress.State
          + "," + targetCompany.PrimaryAddress.Country
      return anEmployeeSummary
    } else {
      return null
    }
  }

  //////    Helper Methods    //////

  /**
   * Method takes taxID and returns company object
   */
  @Param("taxID", "Company tax ID")
  @Returns("Finds company by tax id. Returns type ABCompany")
  private function findCompanyByTaxID(taxID: String): ABCompany {
    // validate all input params sent in by the external system
    if (taxID == null or taxID.Empty) {
      throw new IllegalArgumentException("Invalid input parameter, taxID is null or empty!")
    } else {
      var queryObj = Query.make(ABCompany)
      queryObj.compare(ABCompany#TaxID, Relop.Equals, taxID)
      var resultObj = queryObj.select().AtMostOneRow
      return resultObj
    }
  }
}
```

5. **Deploy code changes.**

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

6. **Perform verification steps.**