Lesson 9 SOAP Web Services

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

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

Exercise 1: Consume a SOAP web service

Exercise

Succeed Insurance needs to connect to a WSI web service hosted on an external system. The web service API offers a method that takes a VIN number as a string argument and returns a string that contains the vehicle's color, year, make, and model.

9.1.1 Requirements

Spec 1 The web service collection package should be called vehiclevin.

Spec 2 ExternalApp's VehicleWsiAPI is accessible at: http://localhost:8890/ab/ws/externalapp/webservice/VehicleAPI?WSDL

Spec 3 Connect to the web service using HTTP authentication with externalappuser/gw as the username/password.

Spec 4 Specify a parameter that limits server response time to 30000 ms.

9.1.2 Tasks

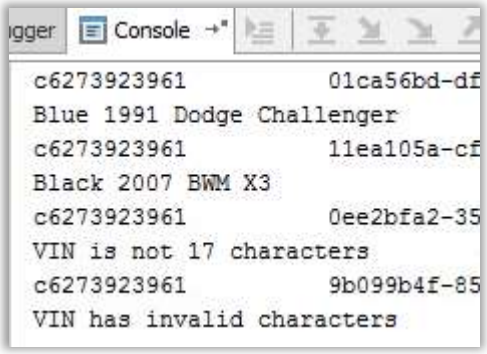
1. **Create a new package called webservice.vehicle.**
2. **Create a web service collection to access the VIN service called vehiclevinwsc.**
3. **Deploy code changes.**
4. **Perform verification steps.**

9.1.3 Verification steps

1. **In Gosu Scratchpad, execute code that calls the external system web service to verify each VIN number in the following table:**

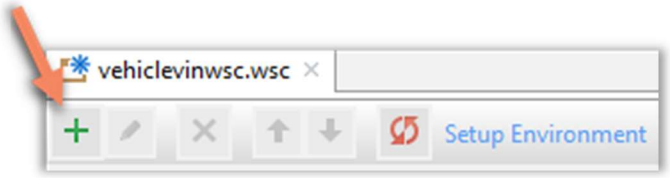
Vehicle Identification Number (VIN)	Result
12345678901234567	Blue 1991 Dodge Challenger
ABCDEFGHIJKLMNOPQ	Black 2007 BMW X3
111	Vin is not 17 characters
$$$$$$$$$$$$$$$	Vin has invalid characters

2. **Verify the output.**



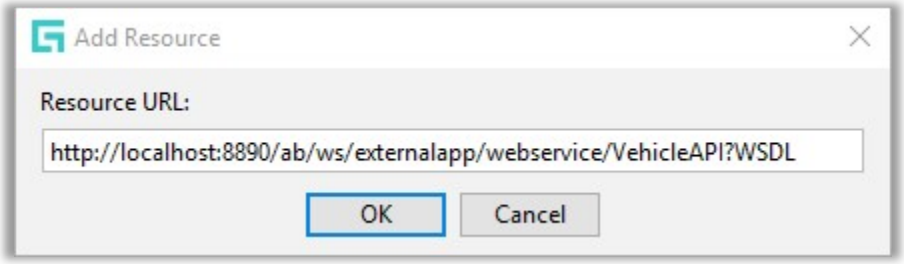
Solution 1: Consume a SOAP web service

1. **Create a new package called webservice.vehicle.**
   a) Right-click on **si.ta** package and select **New ➔ Package**.
   b) Enter **webservice.vehicle** as the new package name.
2. **Create a web service collection to access the VIN service.**
   a) Right-click on the vehicle package and select **New ➔ Web Service Collection**.
   b) Enter **vehiclevinwsc** as the new Web Service Collection name.
   c) Click the **Add New Resource** button.



d) Enter the **ExternalApp VehicleAPI WSDL URL**:

http://localhost:8890/ab/ws/externalapp/webservice/VehicleAPI?WSDL



e) Click the **Fetch** button.

3. **Deploy code changes.**

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

4. **Perform verification steps.**

```
uses si.ta.webservice.vehicle.vehiclevinwsc.vehicleapi.VehicleAPI

var api = new VehicleAPI()
// set API properties
api.Config.Http.Authentication.Basic.Username = "externalappuser"
api.Config.Http.Authentication.Basic.Password = "gw"
api.Config.CallTimeout = 30000
// verify output
print(api.verifyVehicle("12345678901234567"))
print(api.verifyVehicle("ABCDEFGHIJKLMNOPQ"))
print(api.verifyVehicle("111"))
print(api.verifyVehicle("$$$$$$$$$$$$$$$$$"))
```

Exercise 2: Publish a SOAP web service

Exercise

Succeed Insurance needs to publish a WS-I web service that will allow an external system to retrieve company contact information. The API must contain three functions that will: verify if a company exists given a tax identification (tax ID), create a contact note, and generate an employee summary.

9.1.4 Requirements

Spec 1 Create a WS-I web service named CompanyAPI in a package called company.

Spec 2 Create a method called doesCompanyExist:

- Verify that a company exists given a tax identification (tax ID) number.
- Define an input parameter for the tax ID as a String.
- Identify if a company exists for the given tax ID.
- For a company that exists for the given tax ID, return true as a boolean value.

Spec 3 Create a method called createContactNote:

- Create a contact note.
- Define two input parameters: String representing a tax ID and String identifying the body of the note.
- If the tax ID does not correspond to a company, the method does nothing.
- If a company exists for the given tax ID, create a note whose ContactNoteType is **general**, whose subject is **External note**, and whose body is specified by the external system as an input parameter.
- Add the note to the company's ContactNotes array and save the changes.
- The function does not return a value.

Spec 4 Create a method called getEmployeeSummary:

- Generate an Employee Summary return object of type EmployeeSummary.
- Define an input parameter for the tax ID as a String.
- If the tax ID does not correspond to a company, the method does nothing.
- If a company exists with the tax ID, return an EmployeeSummary object.
- The EmployeSummary object must contain the following fields:
- Number of employees as an integer.
- Employee score as an integer.
- Headquarters location as a comma-separated string created by concatenating the city, state, and country of the company's primary address.

9.1.5 Tasks

1. **Create a web service package.**
2. **Create EmployeeSummary class return type.**
3. **Create CompanyAPI Gosu class.**
4. **Create CompanyAPI methods.**

Tip

Create a helper method called **findCompanyByTaxID** that performs a company query.

5. **Deploy code changes.**
6. **Perform verification steps.**

9.1.6 Verification steps

1. **Obtain the CompanyAPI WSDL URL.**

   a) Launch a web browser.

   b) Enter the following URL: **localhost:8880/ab/ws**.

   c) Navigate to **Document/Literal Web Services.si.ta.webservice.company** and select **CompanyAPI** web service.

   - Record the URL from browser address field.
   - http://localhost:8880/ab/ws/si/ta/webservice/company/CompanyAPI?WSDL

2. **Launch SoapUI to test the CompanyAPI.**
3. **Create a new SOAP project.**

   a) **File ➔ New SOAP Project**

   b) Enter **CompanyWSC** as the project name.

   c) Enter http://localhost:8880/ab/ws/si/ta/webservice/company/CompanyAPI?WSDL as the initial WSDL.

   d) Select **Create Requests** option.

   e) Click **OK**.

4. **Test the doesCompanyExist API.**

   a) Double-click on Request 1 under:

   CompanyWSC ➔ CompanyAPISoap12Binding ➔ doesCompanyExist.

   b) Enter the following in the request XML:

   - soap1:gw_locale: **en_US**
   - soap1:gw_language: **en_US**
   - soap1:username: **su**
   - soap1:password: **gw**

- com:taxID: **54-0683626**
- Click the submit request arrow to execute the request.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap
   <soap:Header>
      <soap1:traceability_id>?</soap1:traceability_id>
      <soap1:gw_locale>en_US</soap1:gw_locale>
      <soap1:gw_language>en_US</soap1:gw_language>
      <soap1:authentication>
         <soap1:username>su</soap1:username>
         <soap1:password>gw</soap1:password>
      </soap1:authentication>
   </soap:Header>
   <soap:Body>
      <com:doesCompanyExist>
         <!--Optional:-->
         <com:taxID>54-0683626</com:taxID>
      </com:doesCompanyExist>
   </soap:Body>
</soap:Envelope>
```

c) Verify the output.

```
HTTP/1.1 200 OK
Date: Wed, 07 Dec 2022 02:41:35 GMT
Content-Type: application/soap+xml;charset=utf-8
Transfer-Encoding: chunked
Server: Jetty(9.4.35.v20201120)

<?xml version="1.0"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
 <soap12:Header>
  <gwsoap:traceability_id xmlns:gwsoap="http://guidewire.com/ws/soapheaders">?</gwsoap:traceability_id>
 </soap12:Header>
 <soap12:Body>
  <doesCompanyExistResponse xmlns="http://example.com/si/ta/webservice/company/CompanyAPI">
   <return>true</return>
  </doesCompanyExistResponse>
 </soap12:Body>
</soap12:Envelope>
```

5. Test the getEmployeeSummaryAPI.

a) Double-click on Request 1 under:

CompanyWSC ➔ CompanyAPISoap12Binding ➔ getEmployeeSummay.

b) Enter the following in the request XML:

- soap1:gw_locale: **en_US**
- soap1:gw_language: **en_US**
- soap1:username: **su**
- soap1:password: **gw**
- com:taxID: **54-0683626**
- Click the submit request arrow to execute the request.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soa
   <soap:Header>
      <soap1:traceability_id>?</soap1:traceability_id>
      <soap1:gw_locale>en_US</soap1:gw_locale>
      <soap1:gw_language>en_US</soap1:gw_language>
      <soap1:authentication>
         <soap1:username>su</soap1:username>
         <soap1:password>gw</soap1:password>
      </soap1:authentication>
   </soap:Header>
   <soap:Body>
      <com:getEmployeeSummary>
         <!--Optional:-->
         <com:taxID>54-0683626</com:taxID>
      </com:getEmployeeSummary>
   </soap:Body>
</soap:Envelope>
```

c) Verify the output.

```
HTTP/1.1 200 OK
Date: Wed, 07 Dec 2022 02:44:42 GMT
Content-Type: application/soap+xml;charset=utf-8
Transfer-Encoding: chunked
Server: Jetty(9.4.35.v20201120)

<?xml version="1.0"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
 <soap12:Header>
  <gwsoap:traceability_id xmlns:gwsoap="http://guidewire.com/ws/soapheaders">?</gwsoap:traceability_id>
 </soap12:Header>
 <soap12:Body>
  <getEmployeeSummaryResponse xmlns="http://example.com/si/ta/webservice/company/CompanyAPI">
   <return xmlns:pogo="http://example.com/si/ta/webservice/company">
    <pogo:EmployeeScore>0</pogo:EmployeeScore>
    <pogo:HeadquartersLocation>La Canada,CA,US</pogo:HeadquartersLocation>
    <pogo:NumberOfEmployees>4</pogo:NumberOfEmployees>
   </return>
  </getEmployeeSummaryResponse>
 </soap12:Body>
</soap12:Envelope>
```

6. Test the createContactNote API.

a) Double-click on **Request 1** under:

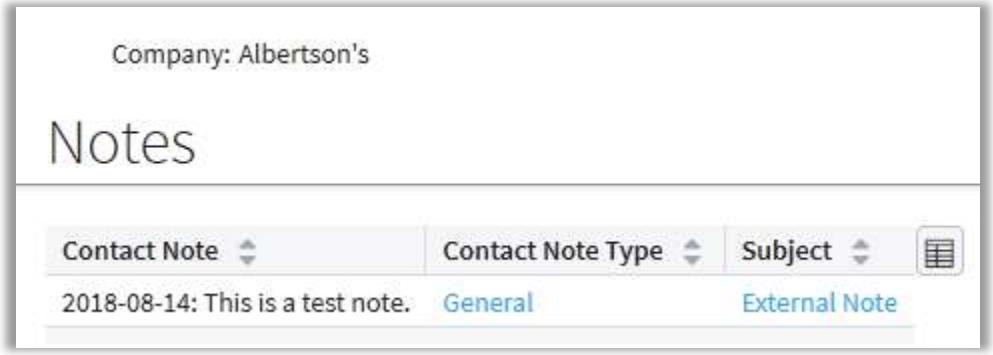
CompanyWSC ➔ CompanyAPISoap12Binding ➔ createContactNote.

b) Enter the following in the request XML:

- soap1:gw_locale: **en_US**
- soap1:gw_language: **en_US**
- soap1:username: **su**
- soap1:password: **gw**
- com:taxID: **54-0683626**
- com:body: **This is a test note**.
- Click the submit request arrow to execute the request.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/so
   <soap:Header>
      <soap1:traceability_id>?</soap1:traceability_id>
      <soap1:gw_locale>en_US</soap1:gw_locale>
      <soap1:gw_language>en_US</soap1:gw_language>
      <soap1:authentication>
         <soap1:username>su</soap1:username>
         <soap1:password>gw</soap1:password>
      </soap1:authentication>
   </soap:Header>
   <soap:Body>
      <com:createContactNote>
         <!--Optional:-->
         <com:taxID>54-0683626</com:taxID>
         <!--Optional:-->
         <com:body>This is a test note</com:body>
      </com:createContactNote>
   </soap:Body>
</soap:Envelope>
```

c) Log in to **TrainingApp** to verify if the note was added to Albertson's.



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.**