Lesson 10 Plugins

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 Super User whose login/password is **su/gw**.

Exercise 1: Create an exchange rate plugin

Exercise

Succeed Insurance needs to retrieve exchange rate values from an external system. The exchange rates shall be retrieved from the external system by clicking on the **Invoke ExchangeRateSet Plugin** button in the UI.

10.1.1 Requirements

Spec 1 Configure the rate exchange plugin so that it will use a custom Gosu class.

Spec 2 The rate exchange plugin must pass authentication parameters to the Gosu class.

Spec 3 Create a custom Gosu class that utilizes the given code for the createExchangeRateSet method.

10.1.2 Tasks

1. **Create a plugin package.**
2. **Create a custom rate exchange Gosu class using best practice naming convention.**

   a) Implement IExchangeRateSetPlugin and InitializablePlugin interfaces and methods.

   b) Copy the following code for the createExchangeRateSet method:

```
override function createExchangeRateSet() : ExchangeRateSet {
  // Create and initialize new exchange rate set
  var erSet = new ExchangeRateSet()
  erSet.Name = "Lab ExchangeRateSet " + gw.api.util.DateUtil.currentDate()
  erSet.Description = "Lab ExchangeRateSet"
  erSet.MarketRates = true
  erSet.EffectiveDate = gw.api.util.DateUtil.currentDate()
  // Create external web service object and set API properties
  var CurrencyAPI = new trainingapp.webservice.currency.exchangeratewsc.currencyapi.CurrencyAPI()
  CurrencyAPI.Config.Http.Authentication.Basic.Username = _username
  CurrencyAPI.Config.Http.Authentication.Basic.Password = _password
  var baseCurrencies = Currency.getTypeKeys(true)
  var priceCurrencies = Currency.getTypeKeys(true)
  // For each base/price currency pair, get exchange rate and add it to set
  for (currentBaseCurrency in baseCurrencies) {
    for (currentPriceCurrency in priceCurrencies) {
      var newExchangeRate = new ExchangeRate()
      newExchangeRate.BaseCurrency = currentBaseCurrency
      newExchangeRate.PriceCurrency = currentPriceCurrency
      newExchangeRate.Rate = CurrencyAPI.getConversionRate(
```

```
          currentBaseCurrency as java.lang.String,
          currentPriceCurrency as java.lang.String)
      erSet.addToExchangeRates(newExchangeRate)
    }
  }
  return erSet
}
```

c) Create four class variables for authentication parameters.

- Two private static final variables for the plugin parameter names.
- Two private static variables for the authentication values retrieved form plugin registry.

d) Configure the set Parameters property to get plugin parameters.

3. **Modify the IExchangeRateSetPlugin registry.**

a) Update Gosu class reference.

b) Create two plugin parameters:

- Name = **externalAppUsername** Value = **externalappuser**
- Name = **externalAppPassword** Value = **gw**

4. **Deploy code changes.**

5. **Perform verification steps.**

10.1.3 Verification steps

1. **Start ExternalApp.**

   Launch **Start ExternalApp** shortcut.

2. **Open TrainingApp.**

   Navigate to **Administration ➔ Training:Plugins ➔ Predefined Plugins**, and click on the Invoke **ExchangeRateSet Plugin** button.

3. **Verify exchange rates are retrieved.**

   There are two ways to verify exchange rates are retrieved:

   - The **Retrieved On** column shows the date and time the exchange rates were retrieved.
   - The rate for each USD-to-nonUSD currency is a decimal value that goes to at least 4 digits. The whole number part of the first two digits are always the same each time you run the plugin. The third and fourth digits vary based on the current minute. For example, the USD to EUR rate is 0.75XX, where XX is the current minute.

Solution 1: Create an exchange rate plugin

1. **Create a plugin package.**

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

   b) Enter **plugin.exchangerate** as the new package name.

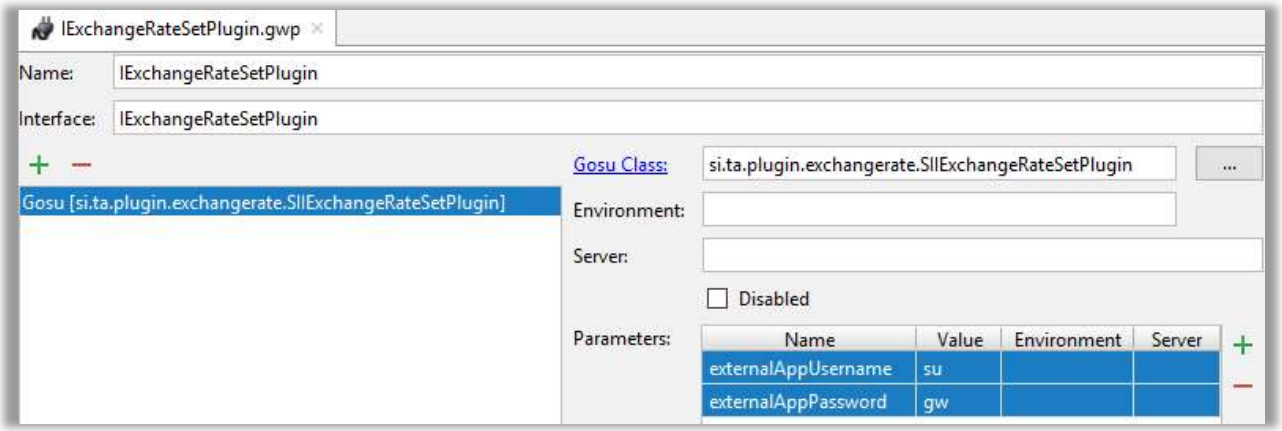
2. **Create a custom rate exchange Gosu class using best practice naming convention.**

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

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

```
package si.ta.plugin.exchangerate

uses gw.plugin.InitializablePlugin
uses gw.plugin.exchangerate.IExchangeRateSetPlugin

class SIIExchangeRateSetPlugin implements IExchangeRateSetPlugin, InitializablePlugin{
  // authentication parameters
  private static final var USERNAME = "externalAppUsername"
  private static var _username : String
  private static final var PASSWORD = "externalAppPassword"
  private static var _password : String

  override function createExchangeRateSet() : ExchangeRateSet {
    // Create and initialize new exchange rate set
    var erSet = new ExchangeRateSet()
    erSet.Name = "Lab ExchangeRateSet " + gw.api.util.DateUtil.currentDate()
    erSet.Description = "Lab ExchangeRateSet"
    erSet.MarketRates = true
    erSet.EffectiveDate = gw.api.util.DateUtil.currentDate()
    // Create external web service object and set API properties
    var CurrencyAPI = new trainingapp.webservice.currency.exchangeratewsc.currencyapi.CurrencyAPI()
    CurrencyAPI.Config.Http.Authentication.Basic.Username = _username
    CurrencyAPI.Config.Http.Authentication.Basic.Password = _password
    var baseCurrencies = Currency.getTypeKeys(true)
    var priceCurrencies = Currency.getTypeKeys(true)
    // For each base/price currency pair, get exchange rate and add it to set
    for (currentBaseCurrency in baseCurrencies) {
      for (currentPriceCurrency in priceCurrencies) {
        var newExchangeRate = new ExchangeRate()
        newExchangeRate.BaseCurrency = currentBaseCurrency
        newExchangeRate.PriceCurrency = currentPriceCurrency
        newExchangeRate.Rate = CurrencyAPI.getConversionRate(
            currentBaseCurrency as java.lang.String,
            currentPriceCurrency as java.lang.String)
        erSet.addToExchangeRates(newExchangeRate)
      }
    }
    return erSet
  }

  override property set Parameters(map : Map<Object, Object>) {
    _username = map.get(USERNAME) as String
    _password = map.get(PASSWORD) as String
  }

}
```

3. **Modify the IExchangeRateSetPlugin registry.**



4. **Deploy code changes.**

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

5. **Perform verification steps.**