Solution: Gosu Classes

TrainingApp needs additional capabilities to help users standardize contact addresses. For example, users need to be able to replace address abbreviations such as "St." with "Street", the unabbreviated form.

In previous labs, you have explored how a configurator can implement a solution using an entity enhancement. For the purposes of this lab, you will create a Gosu class with a static method and then assign this method to the action property of a toolbar button widget.

Requirements

This lab requires that you use TrainingApp 8.0, Guidewire Studio 8.0, and a supported web browser. To view, edit, and delete various contacts, log in to TrainingApp as Alice Applegate. The default URL for TrainingApp is <http://localhost:8880/ab/ContactManager.do>. The login/password for Alice Applegate is aapplegate/gw.

1. Clean up addresses

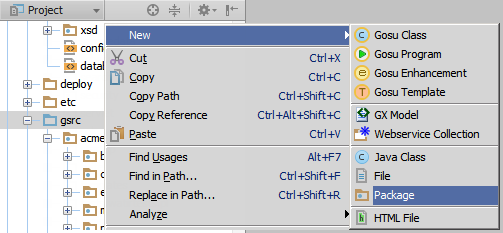
In this exercise, you will first create a class containing a static method that replaces address abbreviations with the unabbreviated word form. You will then add a toolbar button widget to the ABContactAddressesLDV.pcf and assign your static method to the button's action property.

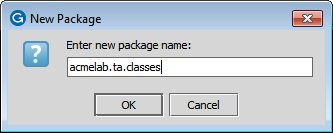
Part A

In Part A of this exercise, you will create the AddressCleanupUtil class and a static method.

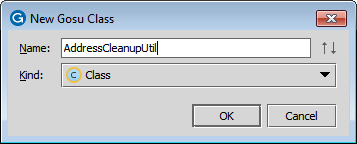
Configuration

1. Open Guidewire Studio for TrainingApp
2. From Studio, if your server is not already running, start the server using Debug 'Server'.
3. Review the Debug console for errors and verify that the application is running in the Debug console.
4. In Guidewire Studio, create the package and class
5. In Guidewire Studio, create a package called acmelab.ta.classes.



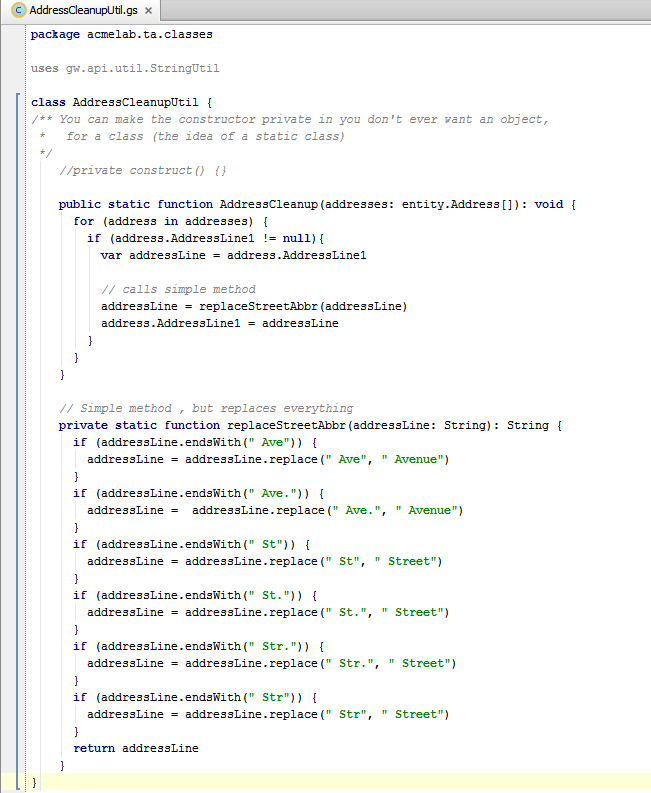


1. Create a Gosu class named AddressCleanupUtil.



1. Create a static method
2. Create a static method in AddressCleanupUtil that defines an array of addresses as the input parameter.
3. For each address in the addresses array, modify the address line 1 field according to the following requirements:

|  |  |
| --- | --- |
| If Address Line 1 ends with… | Replace the value with… |
| Ave | Avenue |
| Ave. | Avenue |
| St | Street |
| St. | Street |
| Str | Street |
| Str. | Street |



package acmelab.ta.classes

uses gw.api.util.StringUtil

class AddressCleanupUtil {

/\*\* You can make the constructor private in you don't ever want an object,

\* for a class (the idea of a static class)

\*/

//private construct() {}

public static function AddressCleanup(addresses: entity.Address[]): void {

for (address in addresses) {

if (address.AddressLine1 != null){

var addressLine = address.AddressLine1

// calls simple method

addressLine = replaceStreetAbbr(addressLine)

address.AddressLine1 = addressLine

}

}

}

// Simple method , but replaces everything

private static function replaceStreetAbbr(addressLine: String): String {

if (addressLine.endsWith(" Ave")) {

addressLine = addressLine.replace(" Ave", " Avenue")

}

if (addressLine.endsWith(" Ave.")) {

addressLine = addressLine.replace(" Ave.", " Avenue")

}

if (addressLine.endsWith(" St")) {

addressLine = addressLine.replace(" St", " Street")

}

if (addressLine.endsWith(" St.")) {

addressLine = addressLine.replace(" St.", " Street")

}

if (addressLine.endsWith(" Str.")) {

addressLine = addressLine.replace(" Str.", " Street")

}

if (addressLine.endsWith(" Str")) {

addressLine = addressLine.replace(" Str", " Street")

}

return addressLine

}

}

**Here's an alternative solution for the Simple method shown above:**

// Reverses the string and uses replace first

private static function replaceStreetAbbreviations(addressLine: String, abbreviations: String[], replace: String): String {

var newAddressLine = addressLine

for (abbr in abbreviations) {

if (newAddressLine.endsWith(abbr)) {

//reverse the string so that we can use replaceFirst rather than replace, which replaces every occurrence

var reverseAddressLine = newAddressLine.reverse()

var reverseAbbr = abbr.reverse()

var reverseReplace = replace.reverse()

reverseAddressLine = reverseAddressLine.replaceFirst(reverseAbbr, reverseReplace)

// reverse it back so it is in the order that it came in

newAddressLine = reverseAddressLine.reverse()

}

}

return newAddressLine

}

1. Deploy your class
2. In Guidewire Studio, reload changed classes.

Hints

**Hint (1)**Use an array for the input parameter for your static method of the type of entity.Address. Recall the syntax for an array as **objectType[]**. For example:

**1 function myFunction(addresses: Address[]): void {**

**2 *// todo***

**3 }**

**Hint (2)**A String in Gosu is itself a class with methods that you can use to find characters in a given string. One helpful method is **contains(*charSequence*)**. Use this method to determine if a string contains a sequence of characters. This method returns a boolean value. For example:

1 **var** string1 = **"This is an example"**

2 **if** (string1.**contains**(**"This"**)) print (**"'This' was found!"**)

3 **if** (string1.**contains**(**"That"**)) print (**"'That' was found!"**)

4 *// Output to Console is 'This' was found*

**Hint (3)**A String in Gosu is itself a class with methods that you can use to replace characters in a given string. One helpful method is the **replace(*charSequence,charSequence*)**. Use this method to find a character sequence (the first parameter) and replace it with another character sequence (the second parameter). For example:

1 **var** string1 = **"This is an example"**

2 **var** string2 = string1.replace(**"is"**,**"xx"**)

3 print (string2)

4 *// Output to Console is "Thxx xx an example"*

In this example's output, observe that xx appears twice.

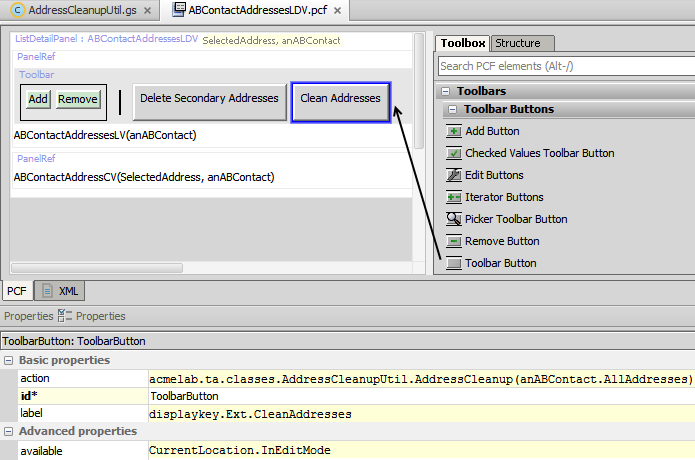
**Hint (4)**Avoid replacing all characters that match a character sequence that are not an abbreviation for a type of street. For example, if AddressLine1 is "555 Straight Stave AVE.", your function should return "555 Straight Stave Avenue" and not "555 Streetaight Street Avenue Avenue.". If AddressLine1 is "444 Ave Maria Stairway St", your function should return "444 Ave Maria Stairway Street" and not "444 Avenue Maria Streetway Street".

Consider using other helpful string methods including reverse(), endsWith(), and replaceFirst().

Part B

In Part B of this exercise, you will add a toolbar button widget to the ABContactAddressesLDV.pcf and assign your static method to the button's action property.

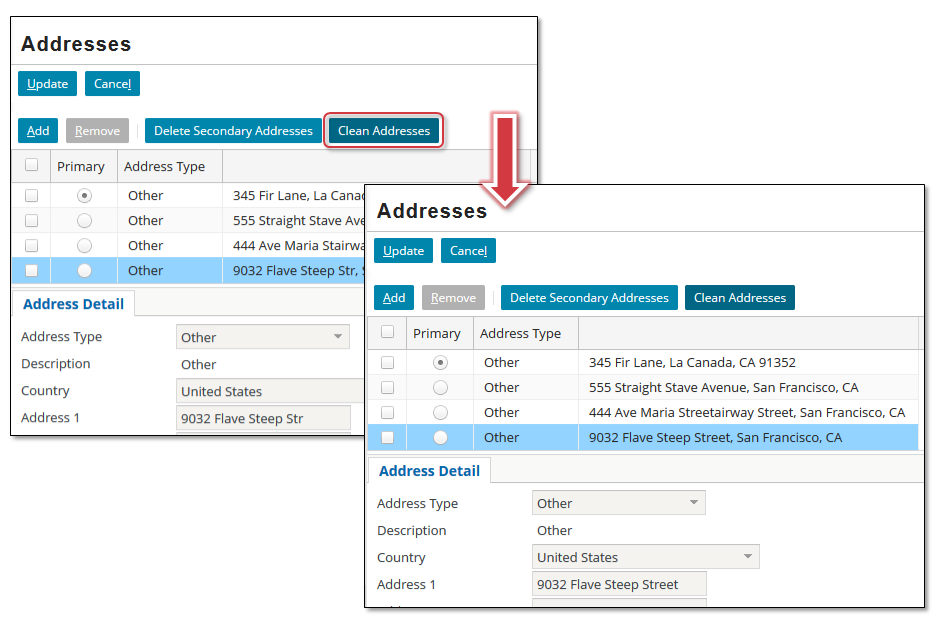
1. Add a toolbar button to the PCF
2. In Guidewire Studio, open ABContactAddressesLDV.pcf .
3. Add a toolbar button to toolbar.
4. Configure the toolbar button
5. Specify an id property for the toolbar button.
6. For the action property, specify the static method.
7. As an argument for the method, use anABContact.AllAddresses. AllAddresses is a derived array of ABContact.
8. Create a new display key for the toolbar button that reads "Clean Addresses".
9. Modify additional widget properties so that the toolbar button is only available when the PCF is in edit mode.



Verification

You will verify how well your function works by first creating addresses[[1]](#endnote-1) and then cleaning them for William Andy.

1. Log in to TrainingApp
2. Log in as Alice Applegate.
3. Deploy the modified PCF
4. In TrainingApp, reload the PCF file changes.
5. Edit the Address for William Andy
6. Open the William Andy contact.
7. In the sidebar menu, click Addresses, then Edit.
8. If not already present, add three new address of the for William Andy:
   * Other , United States , 555 Straight Stave Ave , San Francisco, CA 94104,
   * Other , United States , 444 Ave Maria Stairway St. , San Francisco , CA 94104
   * Other , United States , 9032 Flave Steep Str , San Francisco , CA 94104
9. Click Update.
10. Click Edit and then click Clean Addresses.
11. Verify the following results in the list view detail panel:
    * 555 Straight Stave Avenue, San Francisco, CA
    * 444 Ave Maria Stairway Street, San Francisco, CA
    * 9032 Flave Steep Street, San Francisco, CA
12. Click Update to save your changes.



|  |  |
| --- | --- |
|  | Stop and ask your instructor to review your completed lab. |

1. [↑](#endnote-ref-1)