Lab 13-1

Lab – Embedding a REST Service in a Robot

In this lab, you will return data for all HardyHardware stores existing within a state you choose. This will be CORRECT address information from our Post Office website and will contain latitude and longitude. We will access multiple sources to get this data: the database table you created in Module 11 that contains City, State and Zip records; the HardyHardware site itself, which allows you to search for store locations; and the Post Office site for the final information you'll return and write to the development database. Some things to know before you start:

- Zip codes belonging to a state already exist in the database you created with your Robot in Module 11. Please note that for this training class, the database only contains zips that relate to HH locations – not all zip codes in the country. This was done for performance reasons because we have a limited amount of time in the class.
- The database contains many zip codes for each individual state. Because you will search by **state**, the Robot you are going to build will use the database information and automatically enter each matching zip code into the HH Search box and submit it. The HH Search function will return ALL stores in a 25-mile radius for the first zip code submitted. That will return a list of addresses. It will then loop through ALL zip codes matching the state you enter. So there will be areas of geographic overlap. The same store addresses will be returned multiple times. But you are going to set up the HH store name (it's a store number) as a primary key, and that number is unique. This will prevent duplicate records in the database table you'll create.
- The Robot will loop through the search results returned, one zip code's worth at a time. That data will be entered into an XML variable and a REST web service (from the Robot you created in Module 12) will be called that uses that data. It connects to the Post Office website, enters the HH information and submits it - which returns the corrected data along with latitude and longitude.
- You will store that data in a table you'll create in the development database.
- Depending on the state you select, this robot can take quite a while to run.

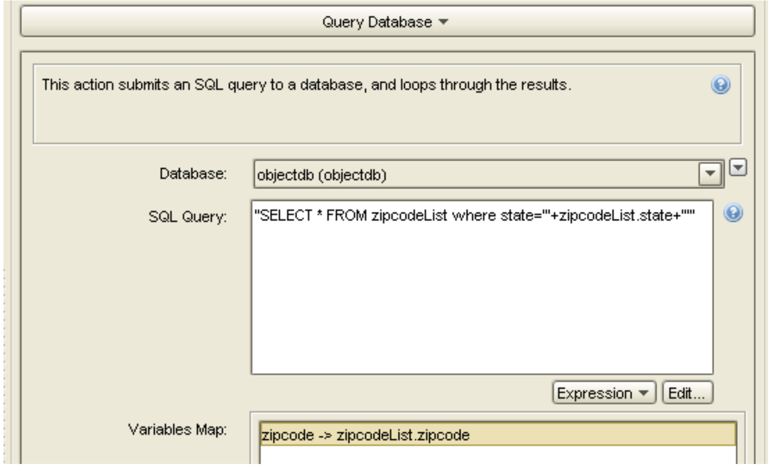
OK, let's get started.

1. Before you create your Robot, perform the tasks as a human user would. Open your web browser and go to http://class.kofax.com/hardyhardware. Notice there is a location search function at the top right of the page.
2. Enter a valid zip code and press [Enter]. The website will return all stores within a 25-mile radius of the zip you entered. Note: If nothing is found, go back and try 18052, a zip code we know has HH stores in the vicinity.
3. Write the address information for the first store returned on a piece of scrap paper.
4. Then go to http://class.kofax.com/Postal/Zipcode.php and enter the information you wrote down.

5. Click submit to view the results. This is the information that you would output to your database for storage.
6. Note that you would have to do this for every store address returned by searching on a single zip code. And every state has multiple zip codes on which to perform a search like this – a daunting task!
7. So…let's build a Robot. Open Design Studio and create a new Robot named "HH_AddressList.robot." For the start URL, enter http://class.kofax.com/hardyhardware.
8. You created the two Types you'll need for this Robot in Modules 11 and 12. Let's use them to create required Variables.
    a. In the Variables panel, add a Variable that uses the StoreAddress Type.
    b. Add a second Variable that uses the zipcodeList Type. Use the checkbox to make it an input variable. And specify "TX" (Texas) as the test value for the state field. Leave the zipcode and city fields blank.
9. The first step in our robot will loop through the zipcodeList table in our development database to match zip codes to the state entered (in our test case,Texas).
    a. Select the end step, right mouse-click and select "Insert Step Before" and "Action Step."
    b. Go to the Action tab of your new step and select "Database" and the "Query Database from the dropdown list.
    c. While still on the Action tab, select "objectdb" as the database.
    d. For the SQL Query, enter:

       "SELECT * FROM zipcodeList where state='"+zipcodeList.state+"'"

       PLEASE NOTE: ALL QUOTES ARE NECESSARY!
    e. For the Variables Map: Click on the "+" symbol to map a new variable.
        i. For Column Name, enter **zipcode**.
        ii. For Variable, enter **zipcodeList.zipcode**.
10. When finished with this step, the Action properties should look like this:



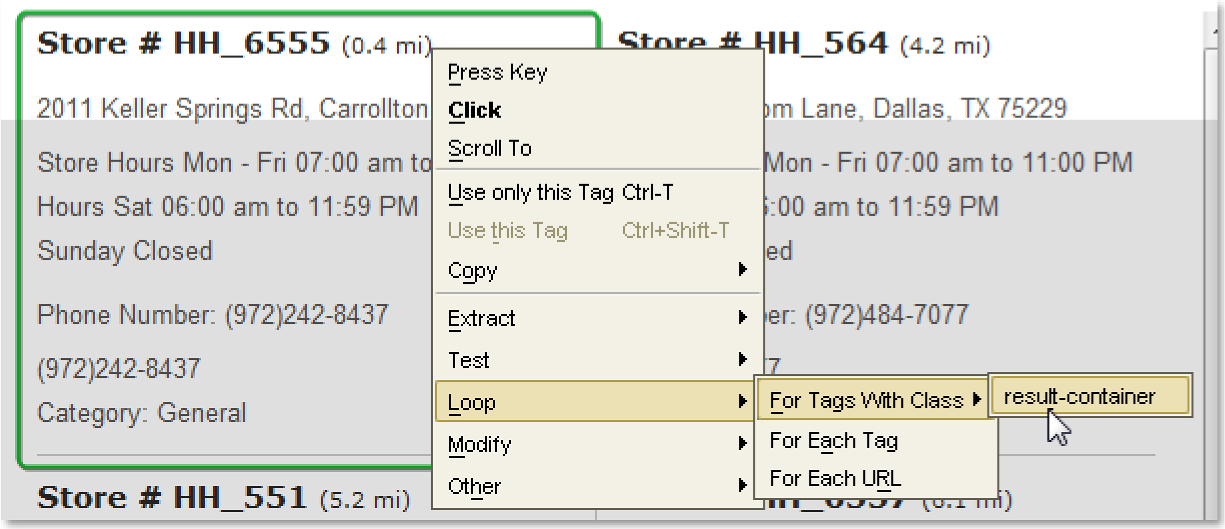
This action submits an SQL query to a database, and loops through the results.

Database: objectdb (objectdb)

SQL Query: "SELECT * FROM zipcodeList where state='"+zipcodeList.state+"'"

Variables Map: zipcode -> zipcodeList.zipcode

11. Test your loop through the database by selecting the end step and clicking on the small forward arrow on the step while looking at the results for the zipcodeList variable in the Variables panel.
12. The next step is to have your robot enter the zip code on the web page. With the end step selected, find the "Search locations" box in the browser panel (scroll over to the right side of the page) and click on it to select it. Right mouse-click and select "Enter Text from Variable" to add the step using the "zipcodeList.zipcode" Variable.
13. Select the "Search" button and add a "Click" step.
14. If you're on the end step in the robot view, you'll see a list of store locations presented in the browser panel (scroll down the page if you don't see them). You'll need to set up a loop step to loop through all the results presented.
    a. Select the first store record. You'll probably have to click on the ↖ ("Select one level out") button to expand the selection for your loop.
    b. Then right mouse-click and select "Loop," "For Tags with Class," "result-container" from the context menu as shown below:



15. Test your loop.
16. You are now going to set up Extract steps using the first record presented for:

- Address
- City
- State
- Zip
- Storename

Notice the first four fields exist on the same line and the Storename (HH_6555) is also contained in one line of content. So you're going to have to set up Converters that use patterns to parse out the data you want. The steps that follow will accomplish that…

a. In your browser panel, select the line containing address, city, state and zip.

b. Right mouse click on the line and select “Extract,” “Text,” and “StoreAddress.address which will create a step called “Extract Address.” Don’t worry about the Converters…yet.

c. Create similar Extract steps for City, State and Zip.

d. Now, go back to each of the four Extract steps you just created and on the “Action” tab of each, add the following converters using “Extraction,” “Advanced Extract” from the dropdown lists:

i.	For Extract Address:	(.*?)(,.*?)	Extract $1
ii.	For Extract City	(.*?,)(.*?)(,.*?)	Extract $2
iii.	For State	(.*,\s+)(..)(.*?)	Extract $2
iv.	For Zip	(.*\s+)(\d{5})	Extract $2

If you wish to try other patterns and expression you might think work better, you may try those as well. Just make sure that they test successfully.

e. Find the Store name in the bowser panel, “Store # HH_6555 (0.4) mi),” and select it. Create a new Extract step. Of course, this selects the entire content and what you want is only “HH_6555.” So you’ll need to add a Converter to parse out this data.

f. For the new “Extract Storename” step, add a Converter using the following pattern: (.*?\#\s)(.*?)(\s.*) and extract $2. You may try creating your own pattern if you wish but make sure to test it by looping through several store addresses.

17. The next step is to place the values extracted into an XML variable. But before you can create the step to do that, you need to have a variable to place the values in. Go to the Variables tab and click the “+” to add a new variable.

18. Name the Variable “requestXML” and choose “XML (simple” as the Type). Then click on the [Edit] button to edit the XML. What you need to enter here is the XML that is generated for a REST Web Service. Remember, we did that in Module 12. If you open the Management Console in your web browser and from the “Repository” | “Robots” tab, select the REST button for the “zipWebService” robot, you can copy the XML code and paste it in the Edit XML window of your new variable. For verification purposes, it is shown below:

```
<?xml version="1.0" encoding="UTF-8"?><rest-request>
 <parameters>
  <variable variableName="StoreAddress">
   <attribute type="text" name="Storename">input text</attribute>
   <attribute type="text" name="Address">input text</attribute>
   <attribute type="text" name="city">input text</attribute>
   <attribute type="text" name="State">input text</attribute>
   <attribute type="text" name="zip">input text</attribute>
   <attribute type="number" name="lat">12.34</attribute>
   <attribute type="number" name="longi">12.34</attribute>
  </variable>
 </parameters>
```

    </rest-request>

19. Click [OK] to close the Edit XML window and [OK] again to close the new Variable window.

20. Now you need to add an Open Variable step. Add an Action Step immediately preceding the end step. Select “Open Variable” from the “Action” dropdown.

21. For the “Variable,” select “requestXML.”

22. If you go to your end step, you’ll see the XML Now you need to set the content of the requestXML Variable to contain the four required values extracted from the HardyHardware web site. This is done similar to an extract step.

    a. If necessary, expand the XML displayed in the browser panel to show the Address, City, State and Zip attributes.

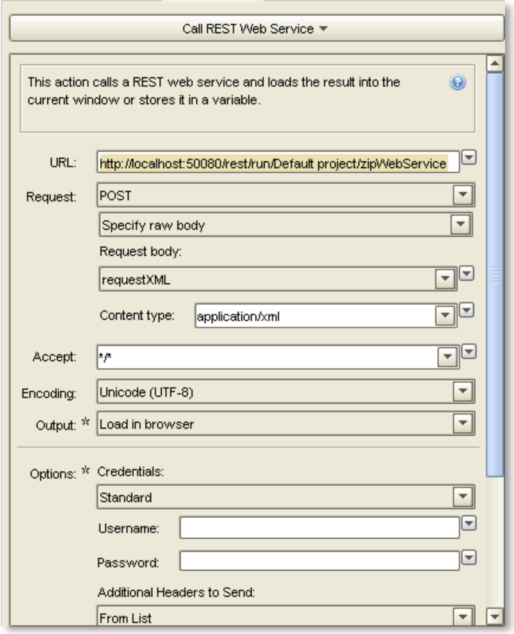
    b. Click on the line that says **<attribute type="text" name="Address">input text</attribute>** to select it. Right mouse-click on the line and from the context menus select “Modify,” “Set” and “Content.” Rename the step, “Set Address.” Make sure you set “New Content” (on the Action tab) to use the “StoreAddress.Address” Variable.

    c. Create similar steps for City, State and Zip.

23. The next step is to call the REST Web Service created from the Robot you built for Module 12. Add a new Action Step preceding the end step. On the Action tab, select the Action dropdown menu and select “Call Web Service” and then “Call REST Web Service” from the menus.

24. Enter the URL of the REST Web Service you tested in Module 12. For your reference, it’s **http://localhost:50080/rest/run/Default project/zipWebService.robot**

25. Set the rest of the properties as follows:



This action calls a REST web service and loads the result into the current window or stores it in a variable.

URL: http://localhost:50080/rest/run/Default project/zipWebService

Request: POST

Specify raw body

Request body:

requestXML

Content type: application/xml

Accept: */*

Encoding: Unicode (UTF-8)

Output: * Load in browser

Options: * Credentials:

Standard

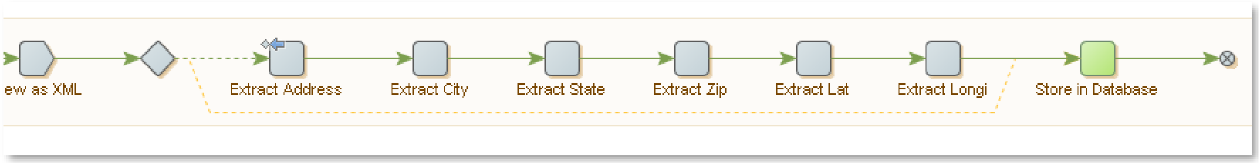
Username:

Password:

Additional Headers to Send:

From List

26. While still on the Action tab of your Call REST Web Service step, scroll to the bottom of the tab and click on the [More] button. Then go to the "Javascript Execution" tab and uncheck the box labeled "Execute JavaScript." Click [OK] to save and close.
27. Finally, you'll need to extract the data which is returned as XML. This is the data you will store in a database. Before you can do that, you need to load and view it as XML. Guess what? There's an Action Step for that! From your end step, insert a New Action Step and select "Load Page" and then "View as XML" from the Action dropdown.
28. If necessary, expand the displayed XML to show the returned information. *NOTE: If the XML code reveals an error, it probably means no result was returned by Post Office website. If that's the case, go back to your "For Each Tag Path" loop and advance it to iteration 2. Then go back to the end step and see if the address information is displayed in the browser window.* Select the line containing the "Address." Select that line, right mouse click on it and select "Extract," "Text" and extract to "StoreAddress.Address."
29. Create similar extraction for:
    a. City
    b. State
    c. Zip
    d. Lat (remember, latitude & longitude will be extracted to Number, not Text)
    e. Longi
30. To accommodate the situation where a matching record is not found and returned by the Post Office website, add a Try step just before the Extract Address step. Then drag the end step node of the new branch to the end node of the main robot. Also set the "Error Handling" of the "Extract Address" step to "Try Next Alternative." Remember to uncheck "API Exception" and "Log as Error." That way if no record is found, extraction will be skipped and the original address returned by the Hardy Hardware website will be output instead.
31. Create a table from the "StoreAddress" type in the "objectdb" database. Remember, you accomplish that from the "Tools" menu dropdown menu.
32. Add a "Store in Database" step. Output to the "objectdb" database and the "StoreAddress" table you just created. The final segment of your Robot should look like this:



33. Save your Robot.
34. Upload to the Management Console.

35. Open the Management Console and create a Kapplet from the Robot called “Store Addresses.” If you don’t remember how to do this, go back to the instructions for Lab 12-3 and review them.
    a. Use the HH_AddressList robot
    b. “State” will be the only value the operator will need to input when the Kapplet is run.
    c. Use the supplied icon for the Robot as you did in Module 12
36. After publishing your Kapplet, install it and then open it.
37. Enter a state (two letter abbreviation) like “AL or PA” and run the Kapplet.
38. After about 5-10 minutes, go to the Data tab in the Management Console and examine the data that is being returned. NOTE: You Robot will continue to run until it has completed, encounters an undiscovered error or until you stop it.