Kofax Kapow 10.3 Training and Certification

Module 13 – Calling a Web Service

Using Results from Other Robots

Kofax Kapow™

KOFAX

Training Module Overview

What we want to accomplish in this training module is this: We'll assume our assignment is to allow a user to write to a database corrected address information, including latitude and longitude, for every HardyHardware store in any given state. We'll create a Robot that uses the database we created in Module 11 along with the REST Web Service we created in Module 12 to accomplish this. What you'll learn or review includes:

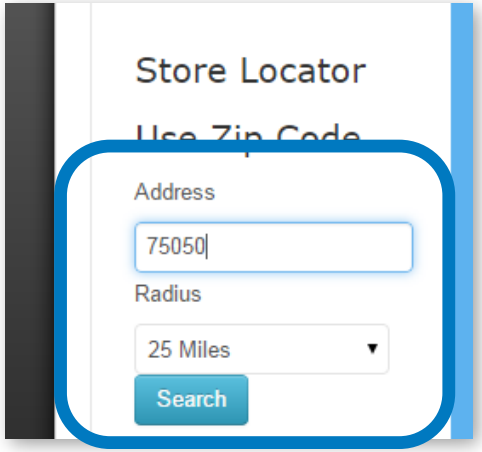
- Providing an Input Variable and Clicking on a Button
- Looping through Tags
- Calling a Web Service
- Opening a Variable
- Setting Content
- Extracting from XML
- Querying and Looping through a database
- Storing Results in another Database

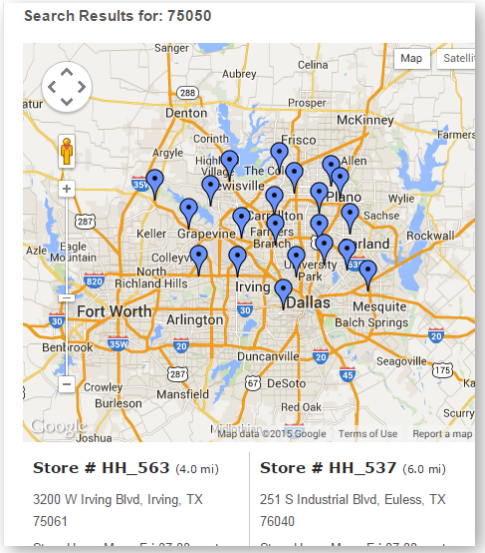
What We've Done So Far

- Sometimes you need to create one Robot to provide the results required by another one.
  - Recall that in Module 11, you created a Robot that took data from an Excel spreadsheet and output it to a database. *(FYI, the Excel spreadsheet we provided to you was created by building two Robots - one that output nearly 42,000 city, state, zipcode, latitude & longitude records and the second that trimmed them down to just over 1300 records that we used in this class.)*
  - And recall that in Module 12, you created a Robot that returned corrected address information with latitude and longitude based on an HH address you looked up from the Post Office website. And you tested it as a REST web service.

To Replicate a Human

- To get all of the HH locations in a state, an operator would have to enter all the zip codes for each HH location, one at a time.
  - After clicking [Search] this would return all stores in a 25-mile radius of that zip code.
  - Of course, many stores would be returned multiple times.
    - But because the Primary Database Key comes from the StoreName field (which is unique), records will not be duplicated by our Robot.





So What Does Our Robot Need to Do?

- Because we already have a database that contains zip codes matched with a state, we can have our new Robot input the zip codes for the entire state by…
  1. querying and looping through the database
  2. matching the state provided for the input
  3. entering the matched zip codes on the webpage
  4. and clicking on the search button

Table: ZIPCODELIST

ZIPCODE	CITY	STATE	Delete
35010	Alexander City	AL	
35064	Fairfield	AL	
35124	Pelham	AL	
35150	Sylacauga	AL	
35173	Trussville	AL	
35210	Birmingham	AL	
35242	Hoover	AL	
35244	Birmingham	AL	
35404	Tuscaloosa	AL	
35501	Jasper	AL	
35630	Florence	AL	
35757	Madison	AL	
35801	Huntsville	AL	
35803	Huntsville	AL	
36066	Prattville	AL	
36117	Montgomery	AL	

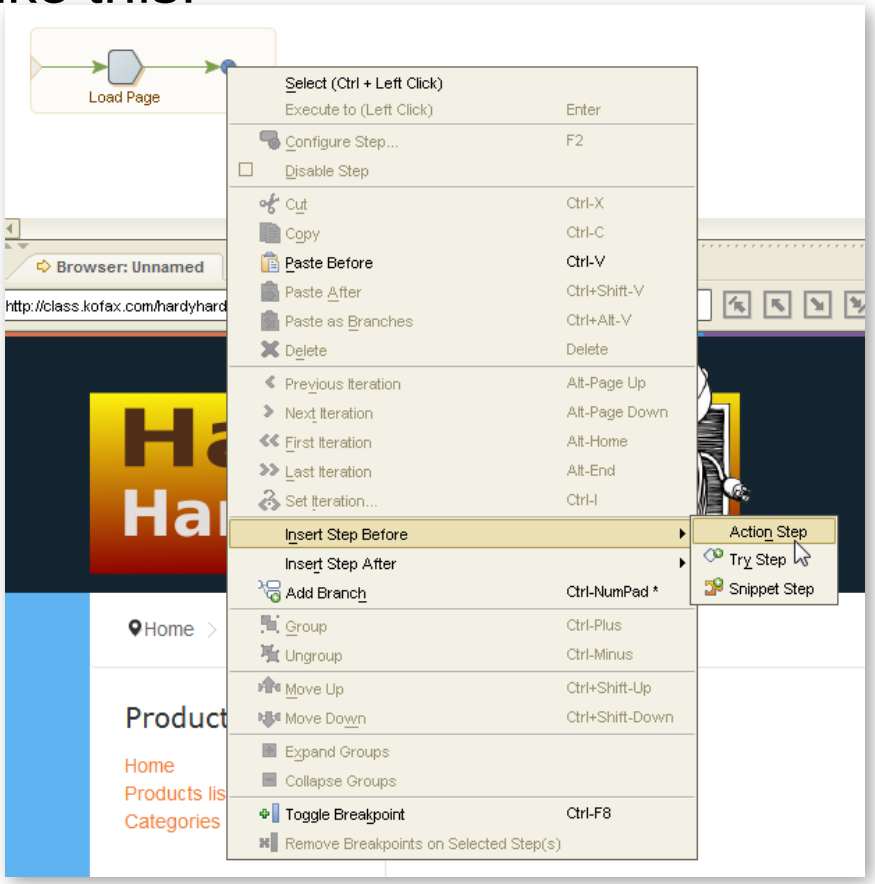
Create New Robot

- OK. It's time to create a new Robot.
  - We will use the StoreAddress and zipcodeList Types we created earlier, so we don't need to create any new Types.
  - However, we do need to add the Variables to our Robot.
  - So we'll create two new Variables, one for each Type. And we'll set the zipcodeList Variable to an Input Variable since it will be providing the input for the store search that a human would normally be manually entering.

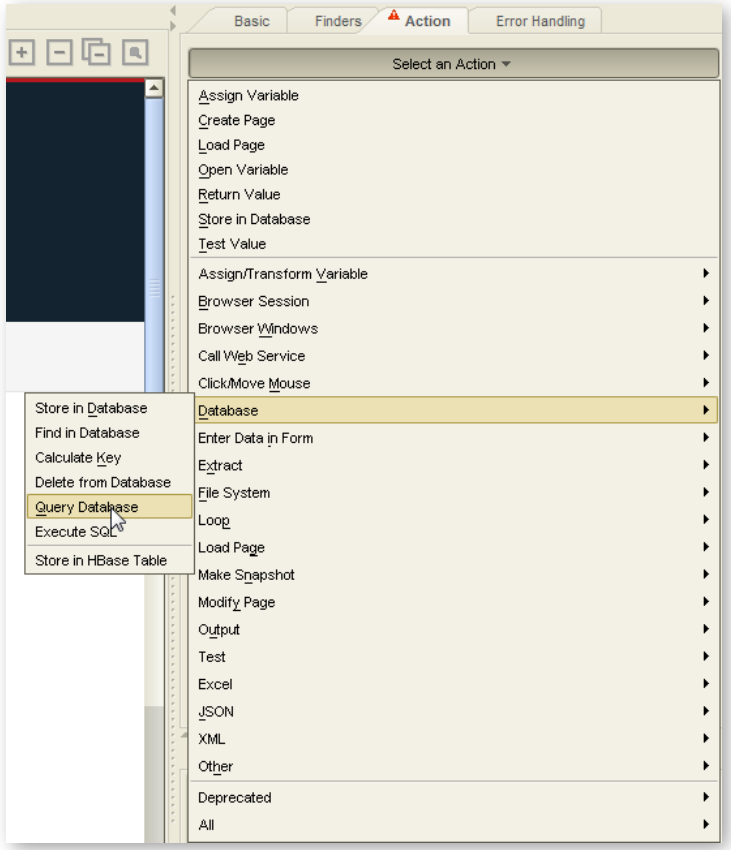
Looping through the Database

- This can be accomplished with a "Query Database" Action step.
- So after loading the home page for the HH website, we'll add a Query Database step like this:

1.

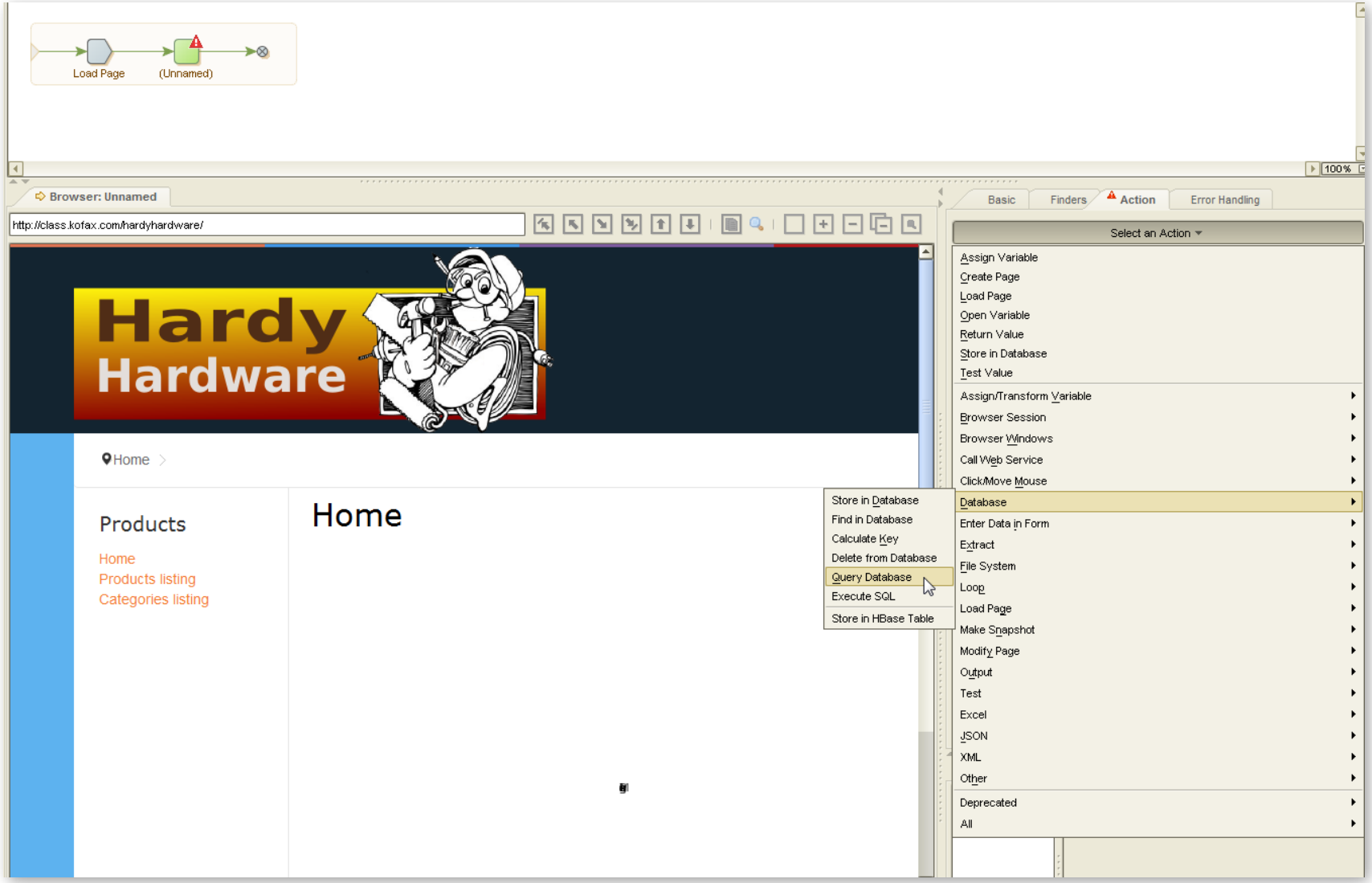


2.



Make Query Database Step

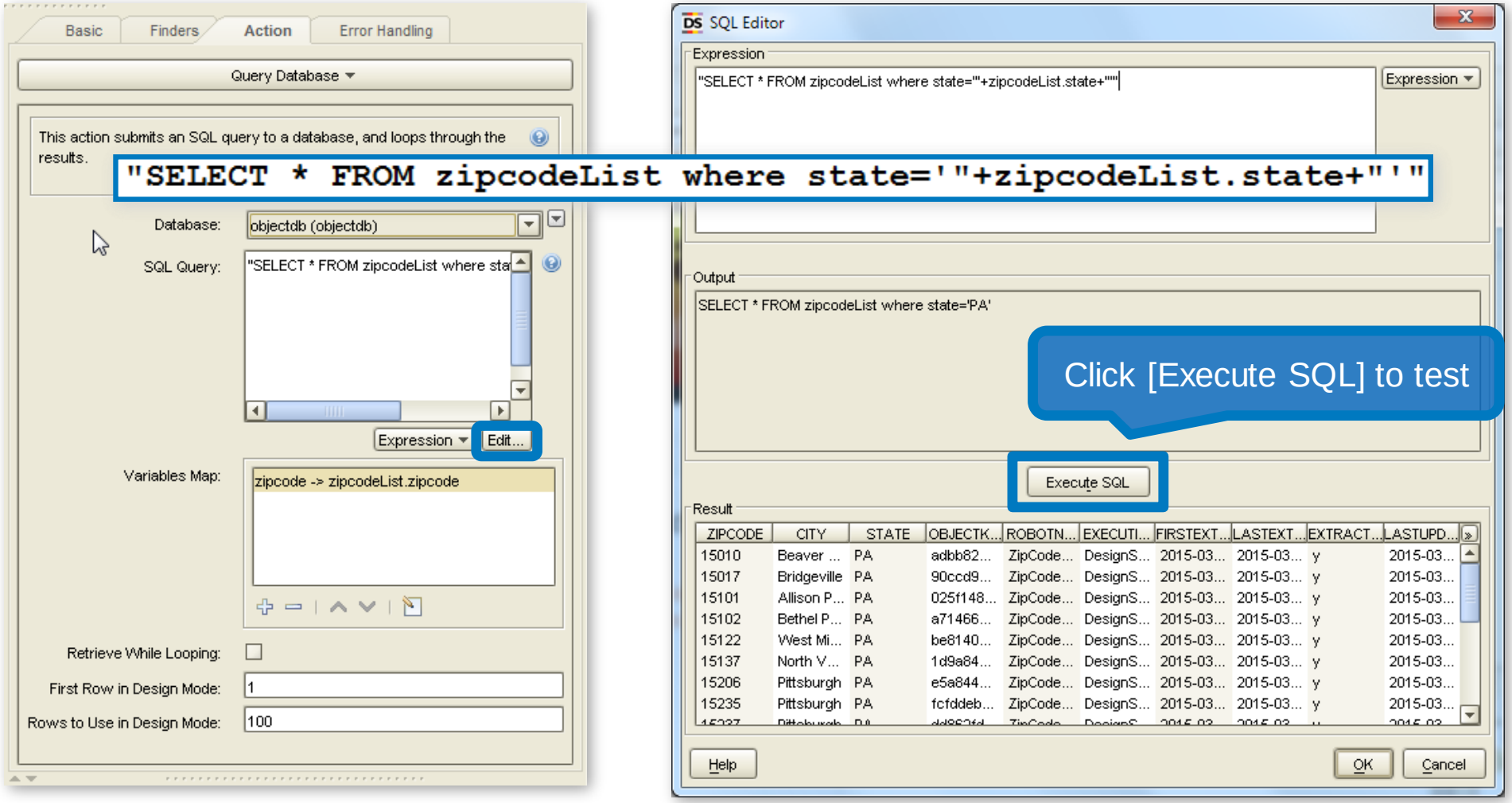
- Select the new unnamed step and select Query Database from the Action dropdown in the step's properties.



What Does a Query Database Step Do?

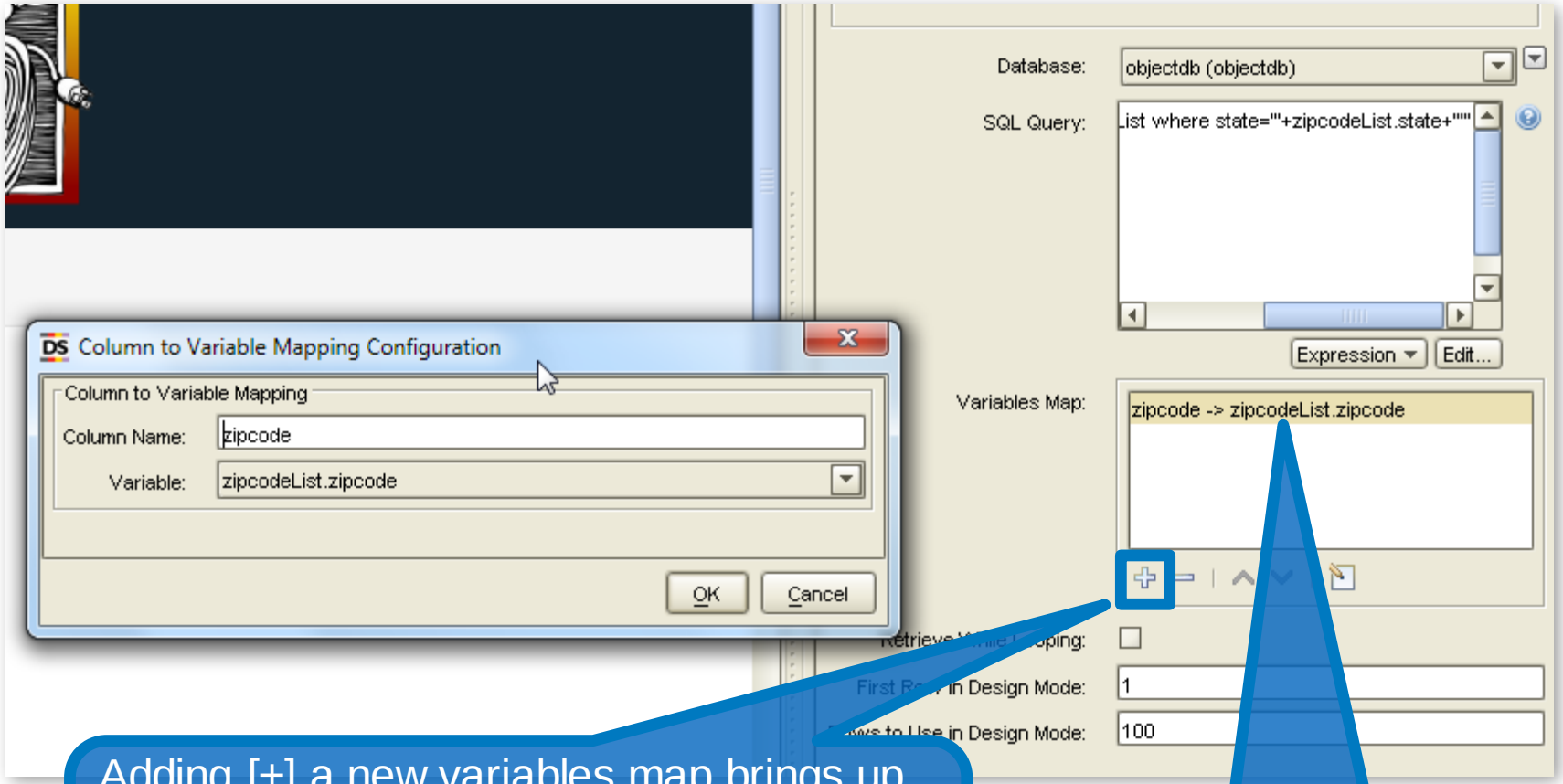
- The Query Database action submits an SQL query to a database and loops through the results. The SQL should be specified using an expression. At each iteration of the result loop, the values of the current row in the result set can be assigned to variables. Properties on the Action tab are as follows:
  - **Database***:* Choose which database to query
  - **SQL Query***:* This field must contain a valid SQL query in the form of an expression. The "Edit" popup dialog allows the SQL query to be tested, showing a sample of the output.
  - **Variables Map***:* Specify the mapping from result columns to variables. Click the plus sign to add a new mapping and the minus sign to remove an existing one. A mapping consists of a column name and a variable name. The column name must match the name of a column returned by the SQL Query, and the variable name is chosen from a list of existing variables. Note, that the type of the column should match the type of the chosen variable. Otherwise, an error may be generated during execution. That is, trying to store a text column in an integer variable will cause an error.

Adding an SQL Query as an Expression



Remember to reference the online Help for assistance on writing expressions.

And Map Your Variables

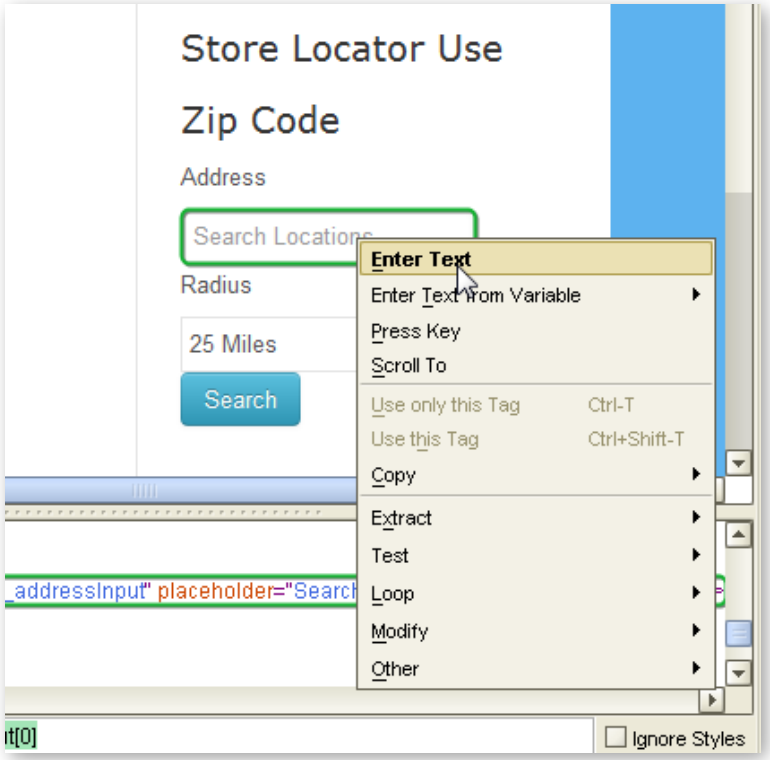


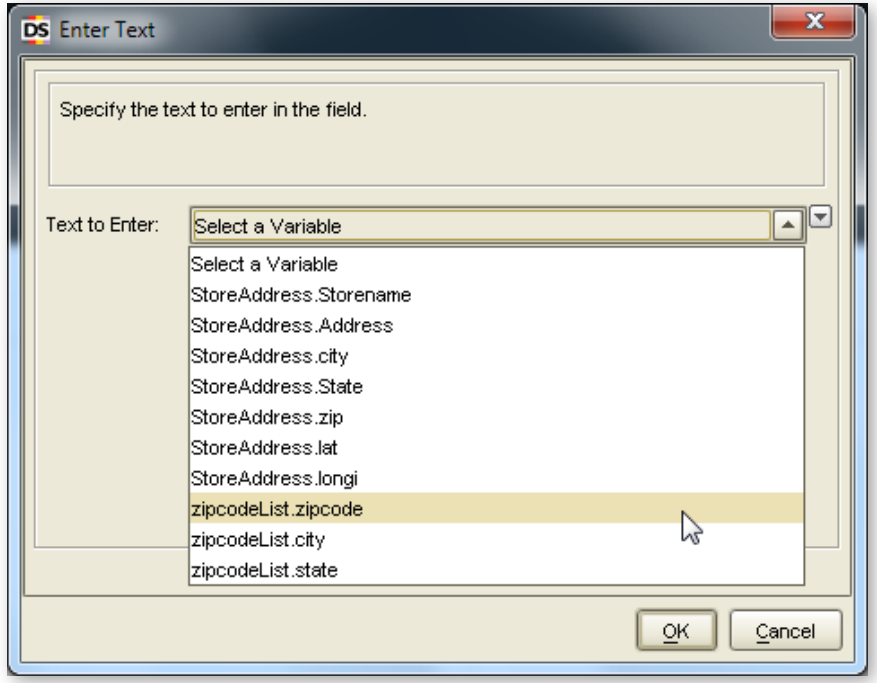
Adding [+] a new variables map brings up the dialog box shown above. We'll map the zipcode column in our database to the zipcodeList.zipcode variable…

…which results in this.

Then Add an Enter Text Step

- If a human were searching the web site, he/she would enter the zip manually…but our Robot will do this from the database providing the value or the zipcodeList.zipcode variable. Below, we're making that happen.

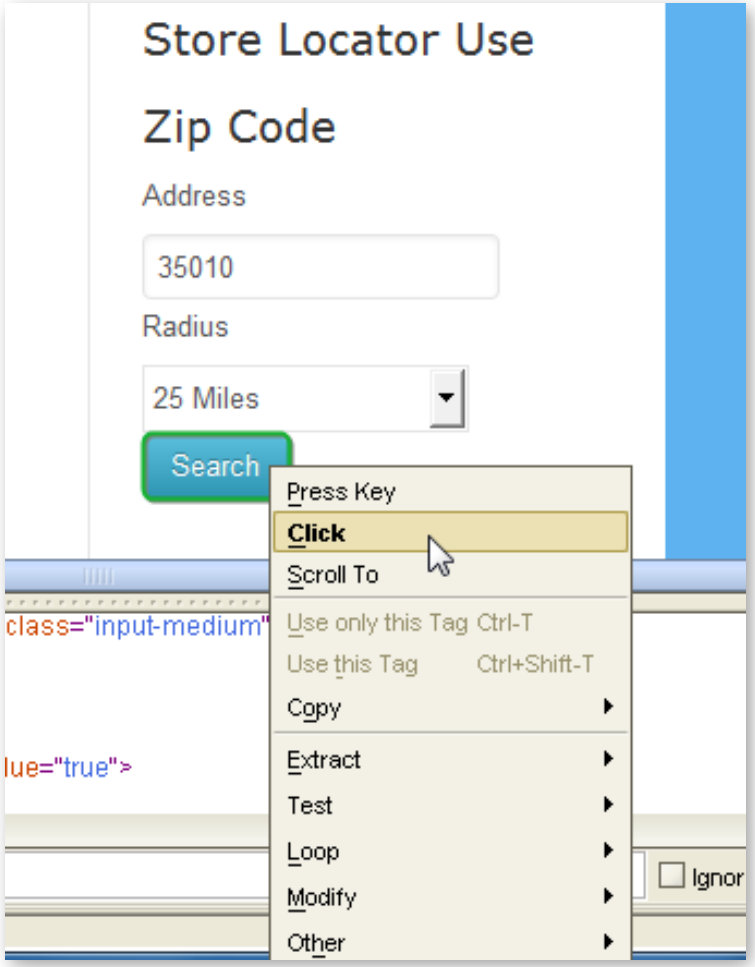




And Add a Click Step to Search

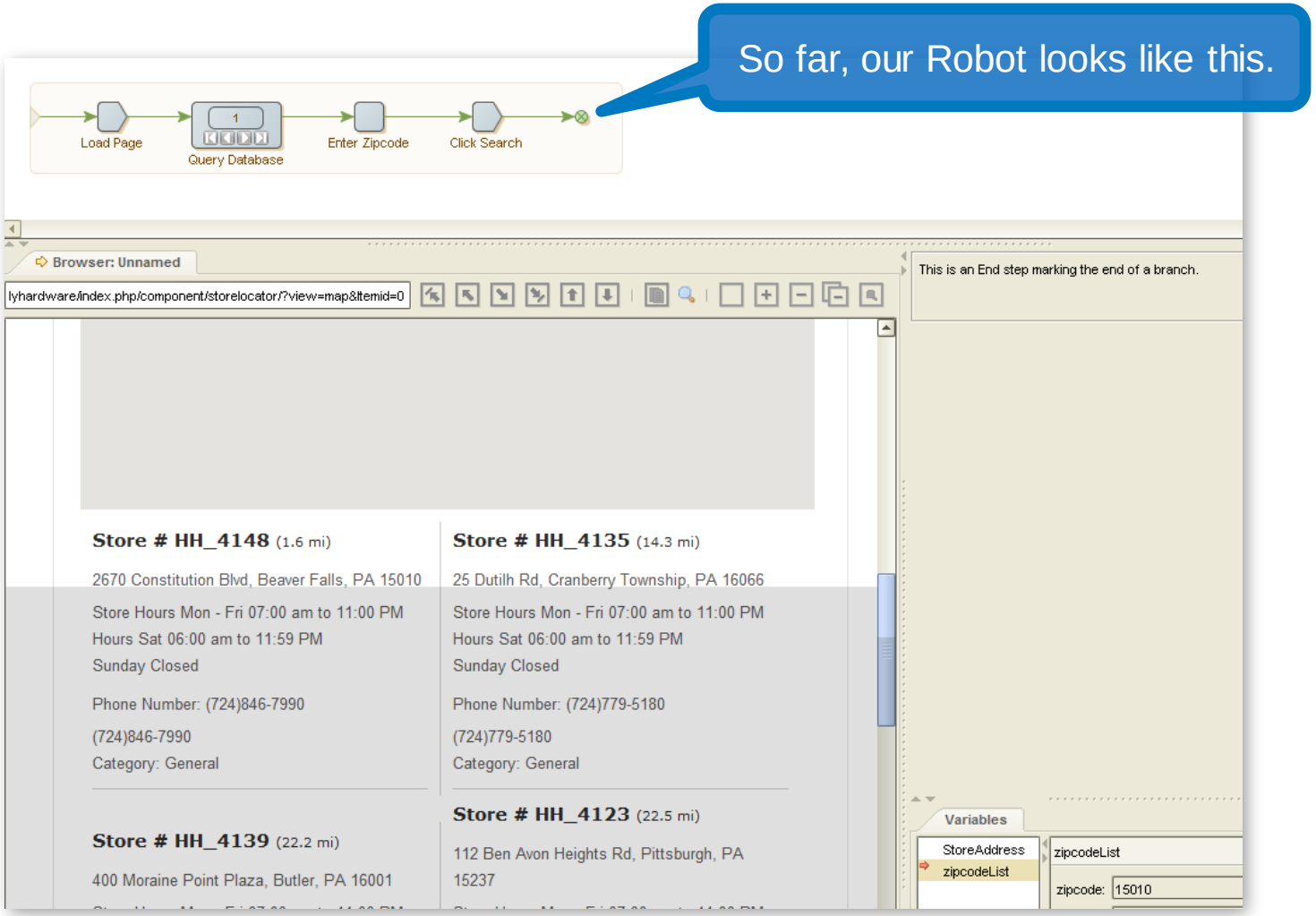
- Once the text is entered, the search button would be clicked to return the values.

- So we'll add a "Click" Action step to accomplish this…

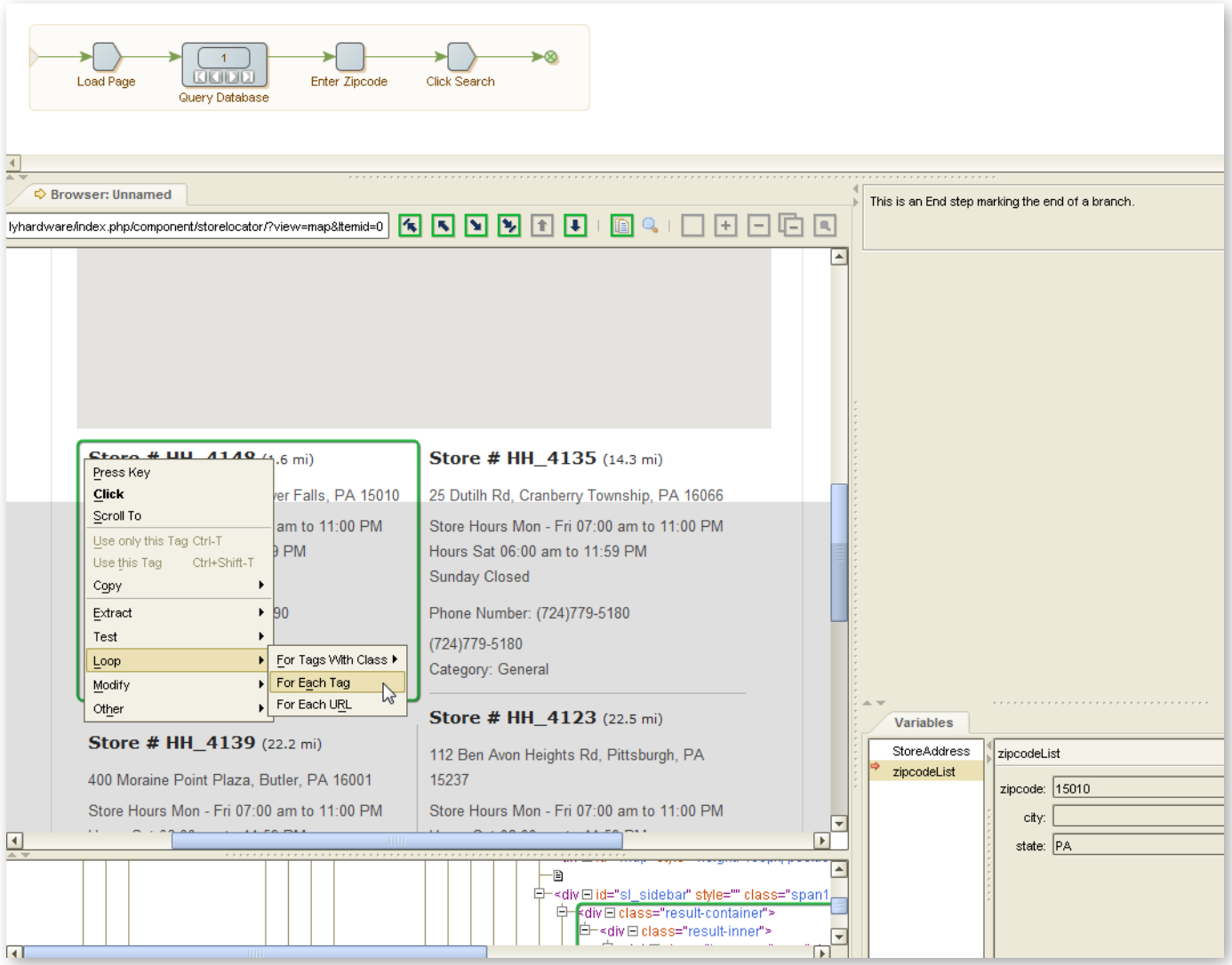


Results Returned

- Results have been returned by the first zip code matching the state in our input variable.
- We will need to loop through the results.

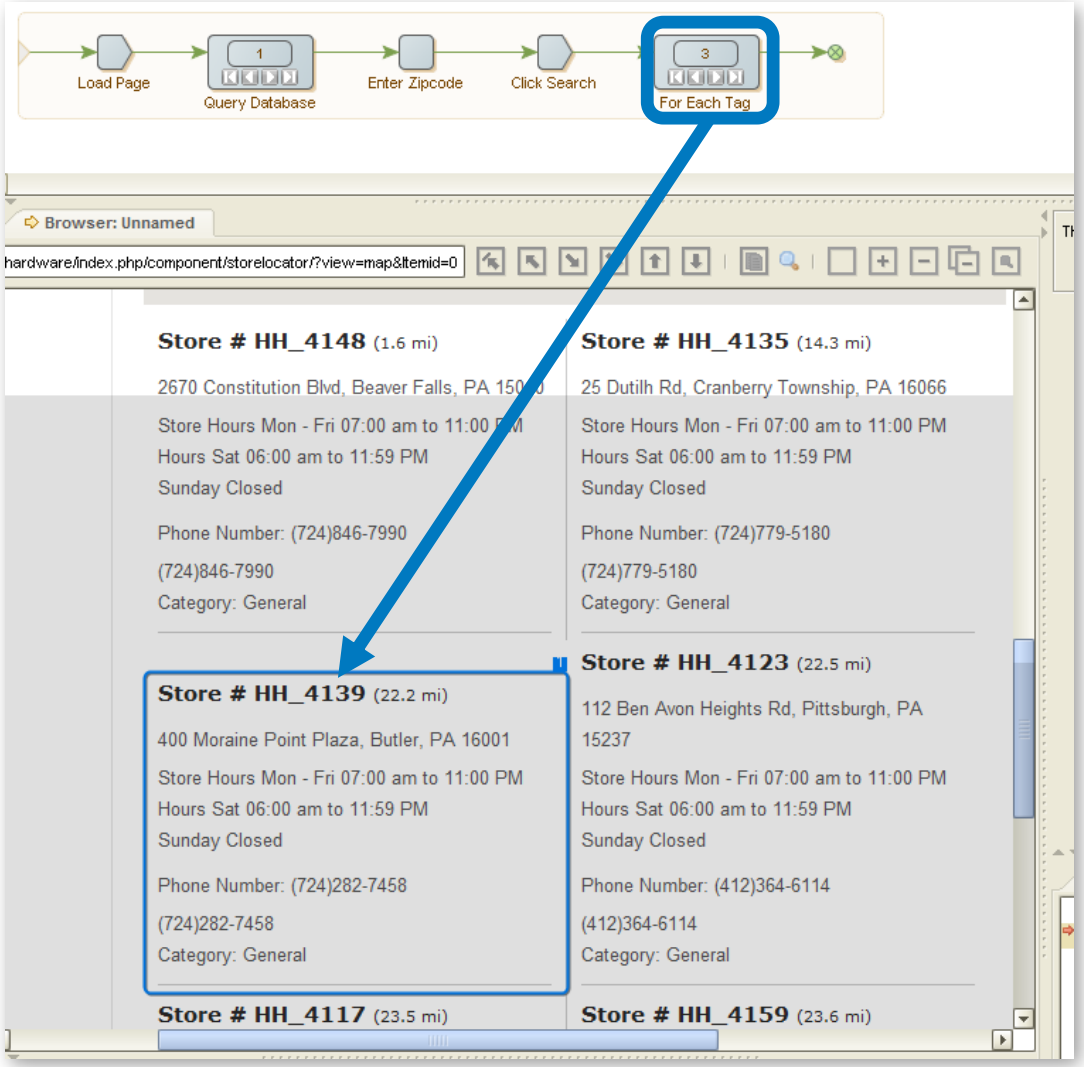


Add a "For Each Tag" Loop



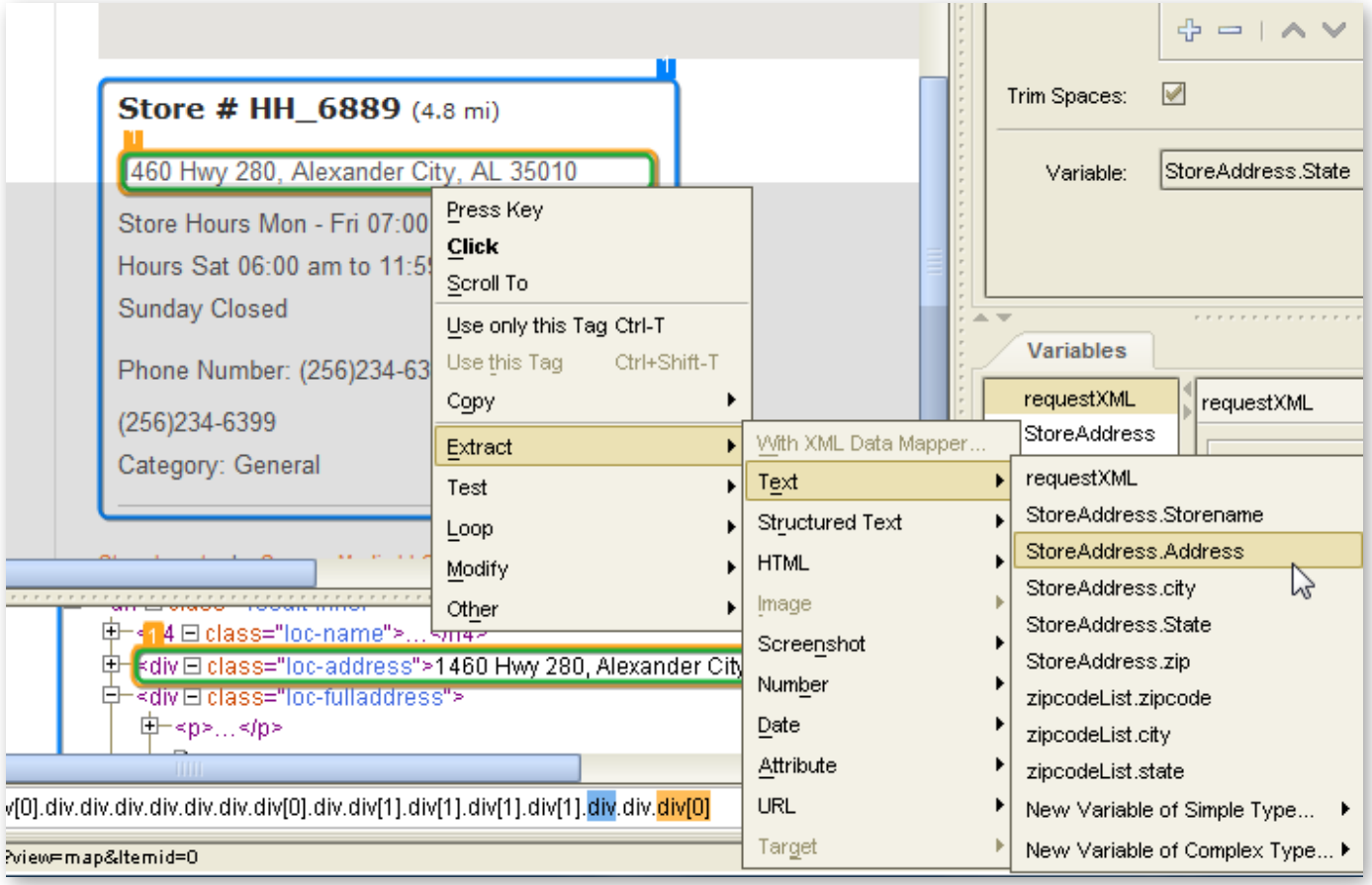
Testing Loop

- Remember to test your loop by passing that step (in this case, go to the end step) and advancing the loop with the arrow button on the loop step.
- In this example, we've advanced to the third item in the loop.



Now We Need to Extract the Address, City, State and Zip

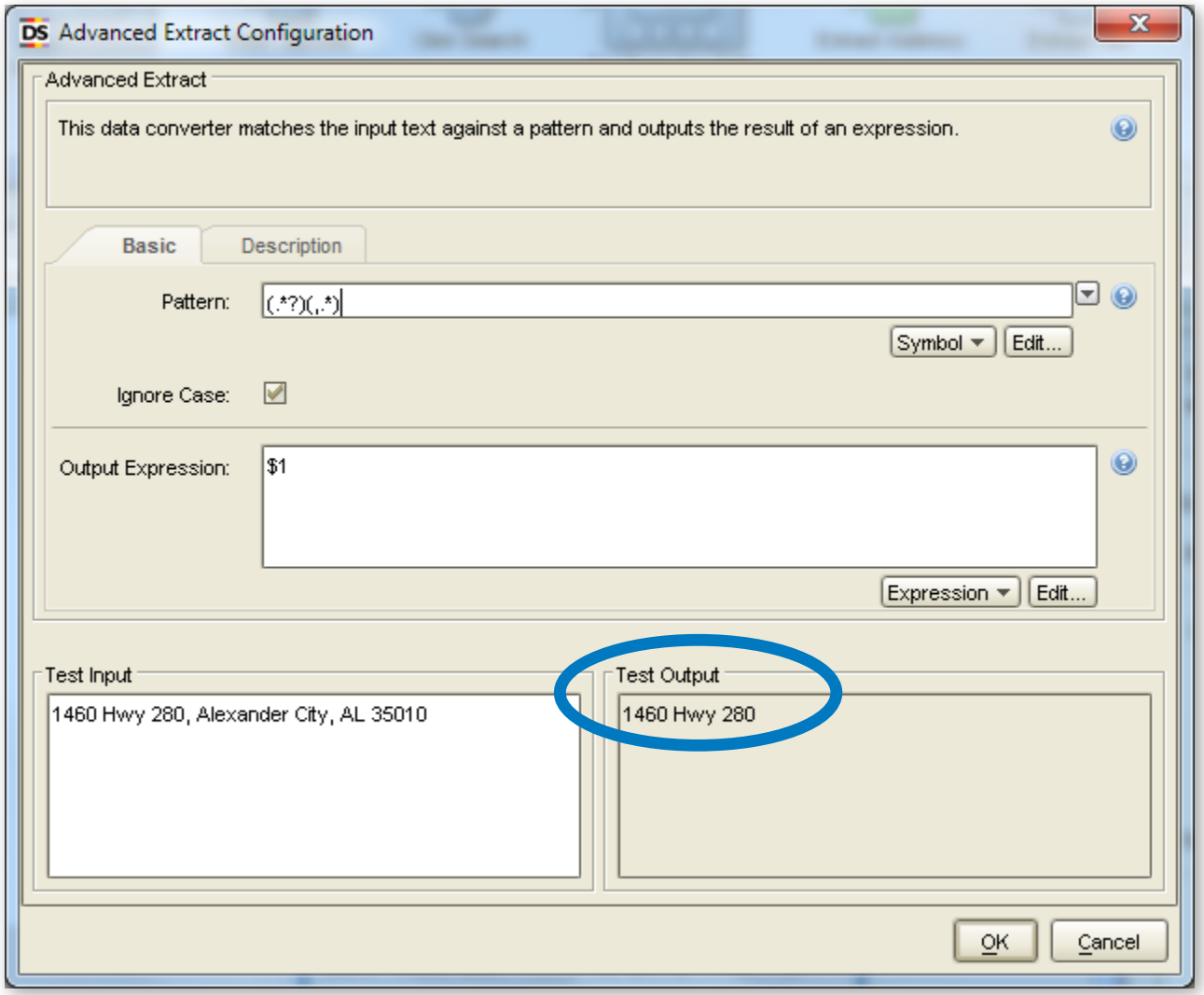
- We'll add four Extract Steps to our Robot.
- All the data for the four fields exists on the same line.



Create Converters using Patterns for Each Field

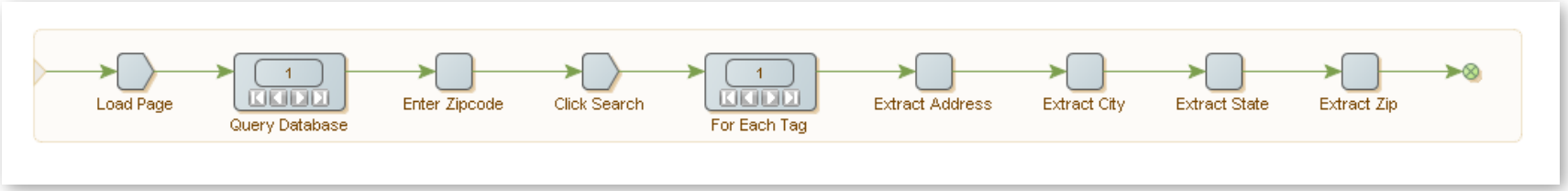
- Text looks like this: **1460 Hwy 280, Alexander City, AL 35010**
  - Pattern for address could be written (.*?)(,.*)
    - $1 would extract everything before the first ","
  - Pattern for city could be written (.*?,\s?)(.*?)(,*.?)
    - $2 would extract everything after the first comma and space (zero or one occurrence) and nothing including and after the second comma.
    - "Collapse and trim spaces" will remove ignore spaces
  - Pattern for state could be written (.*,\s+)(..)(.*?)
    - $2 will extract any two characters after anything that is followed be a comma and one or more occurrences of a space
  - Pattern for zip could be written (.*?\s+)(\d{5})
    - $2 would extract everything after anything after one or more occurrences of a space.

Test Each Pattern as you Go Along



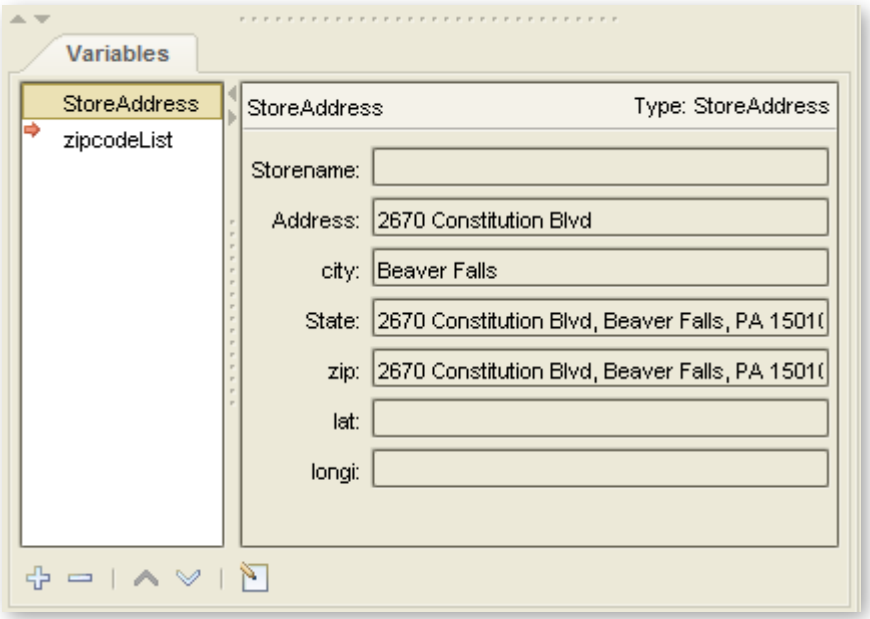
Check your Extraction Before Going On

By selecting the end step and looping, you can ensure your extraction is working the way you want it to.



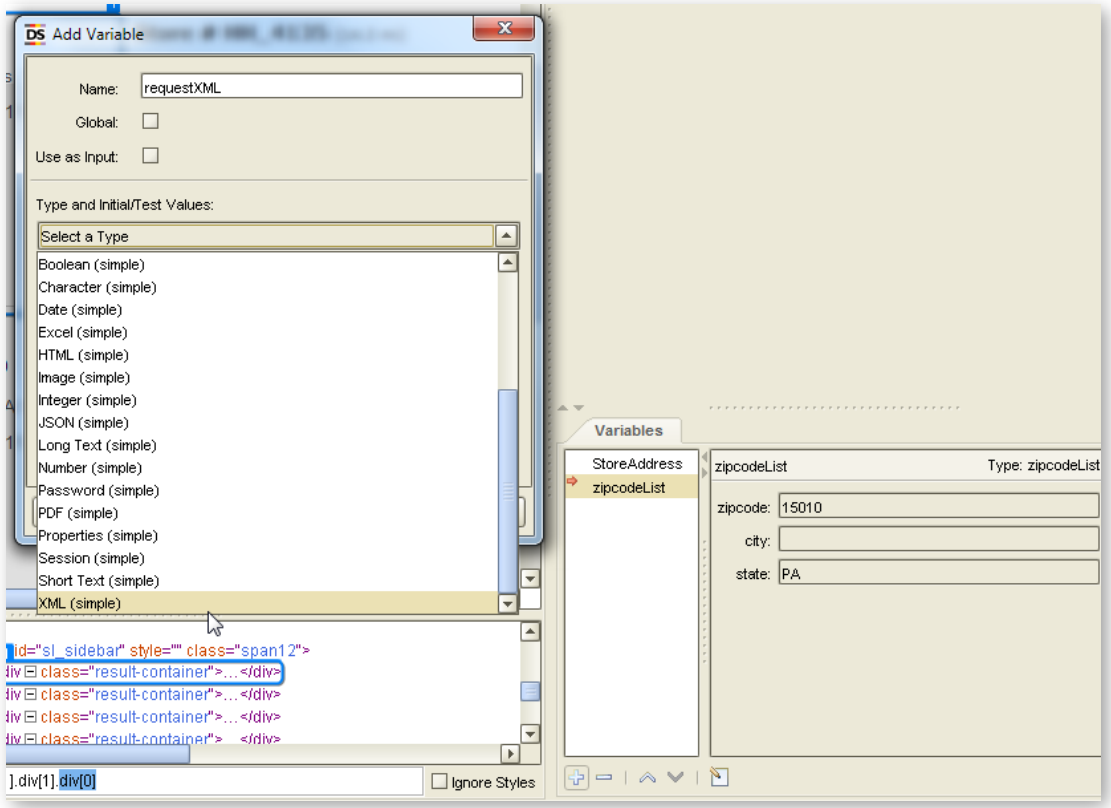
Results are returned to the Variables panel.

The next step will be to replicate the action of an operator who manually enters this information into the Post Office website, which returns corrected address information along with latitude and longitude. We've already built a Robot to accomplish this, which well call as a REST service.

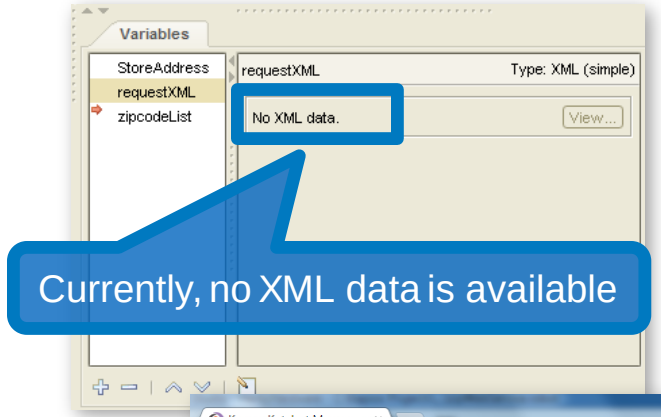


Create a New Variable

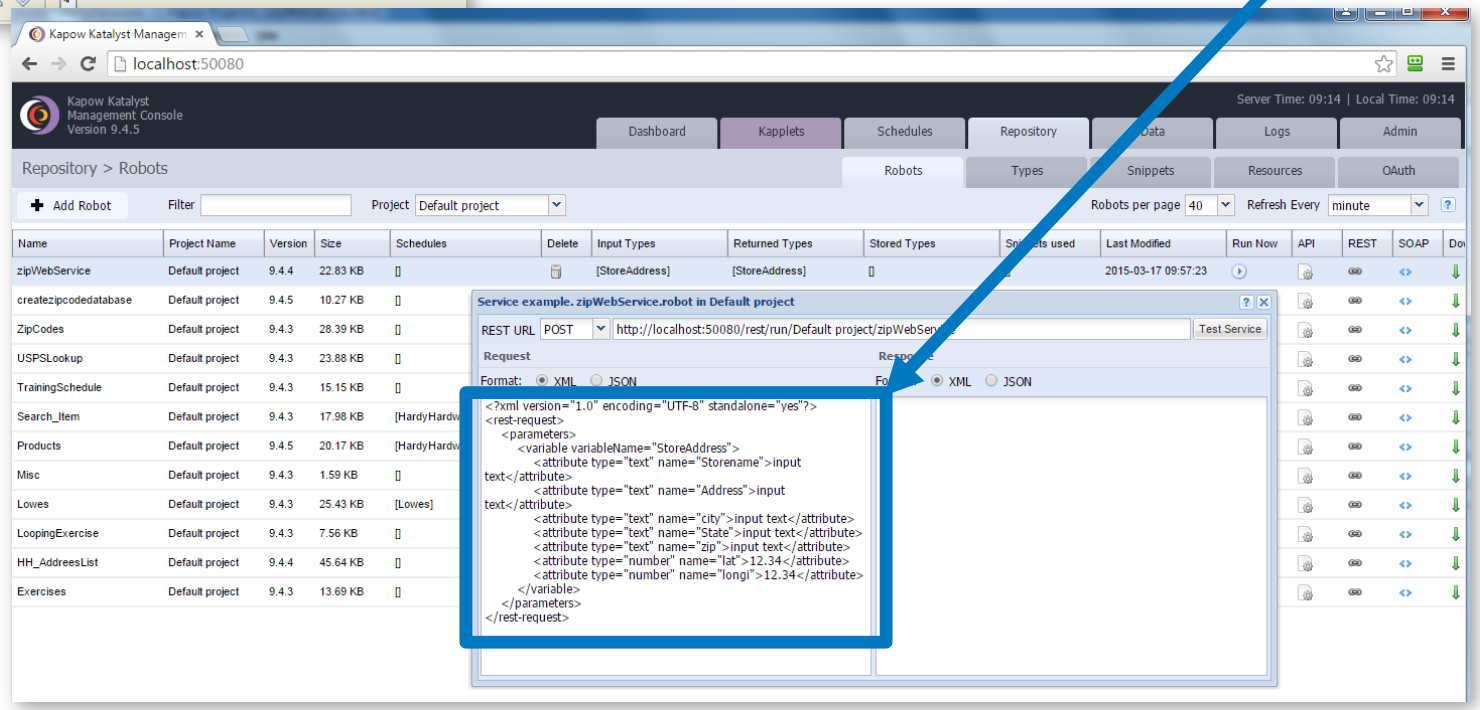
- Because the corrected data we want to extract will be residing in an XML file created by our REST web service, we need a variable to contain that data.
- We'll create a new simple variable called requestXML.



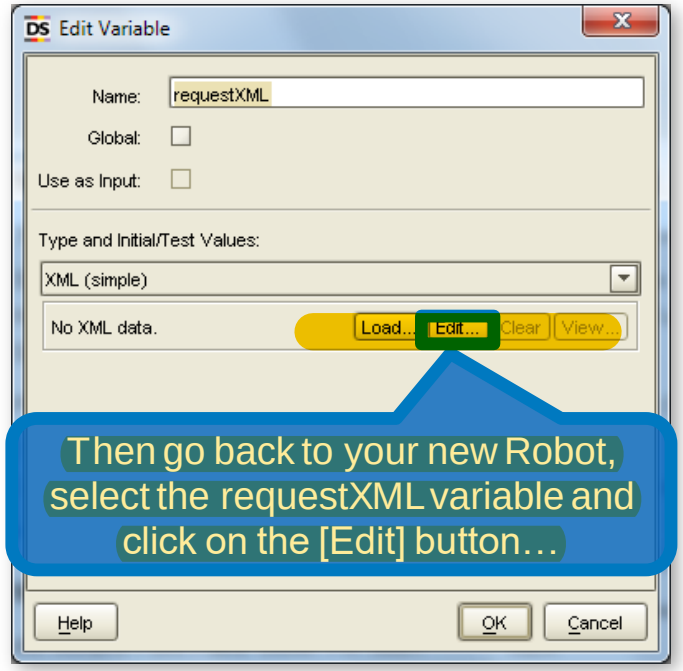
Copy XML Code into the Variable



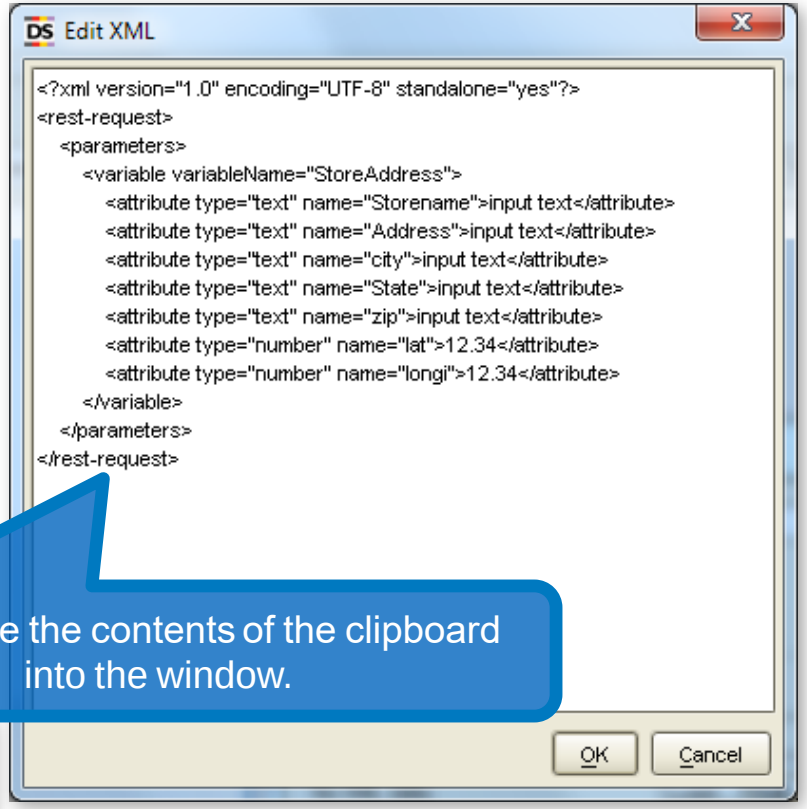
If you recall, we uploaded our "zipcodeService" robot to the Management Console in an earlier module…and we tested it as a REST Service and verified it worked. If you open the Management Console, you can click on the REST button for that robot and view the XML. Copy it to the clipboard.



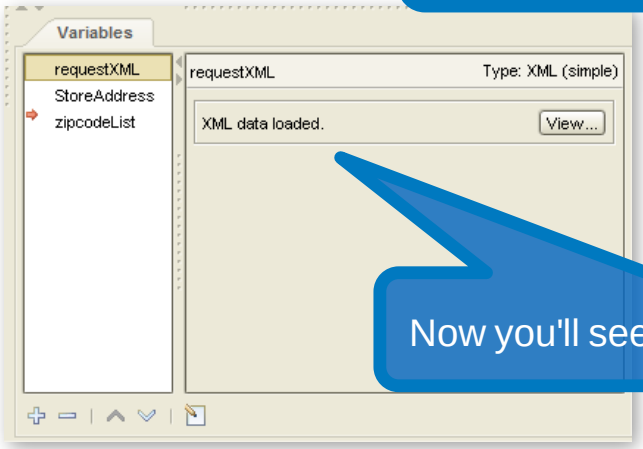
Paste into Variable



Then go back to your new Robot, select the requestXML variable and click on the [Edit] button…



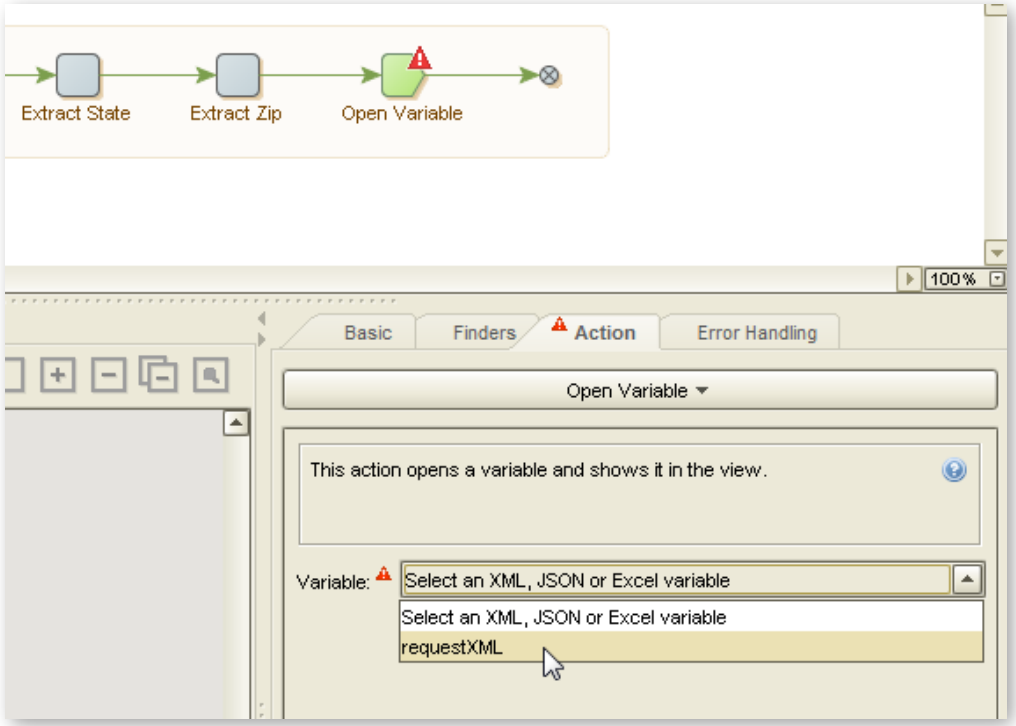
…and paste the contents of the clipboard into the window.



Now you'll see XML data is loaded.

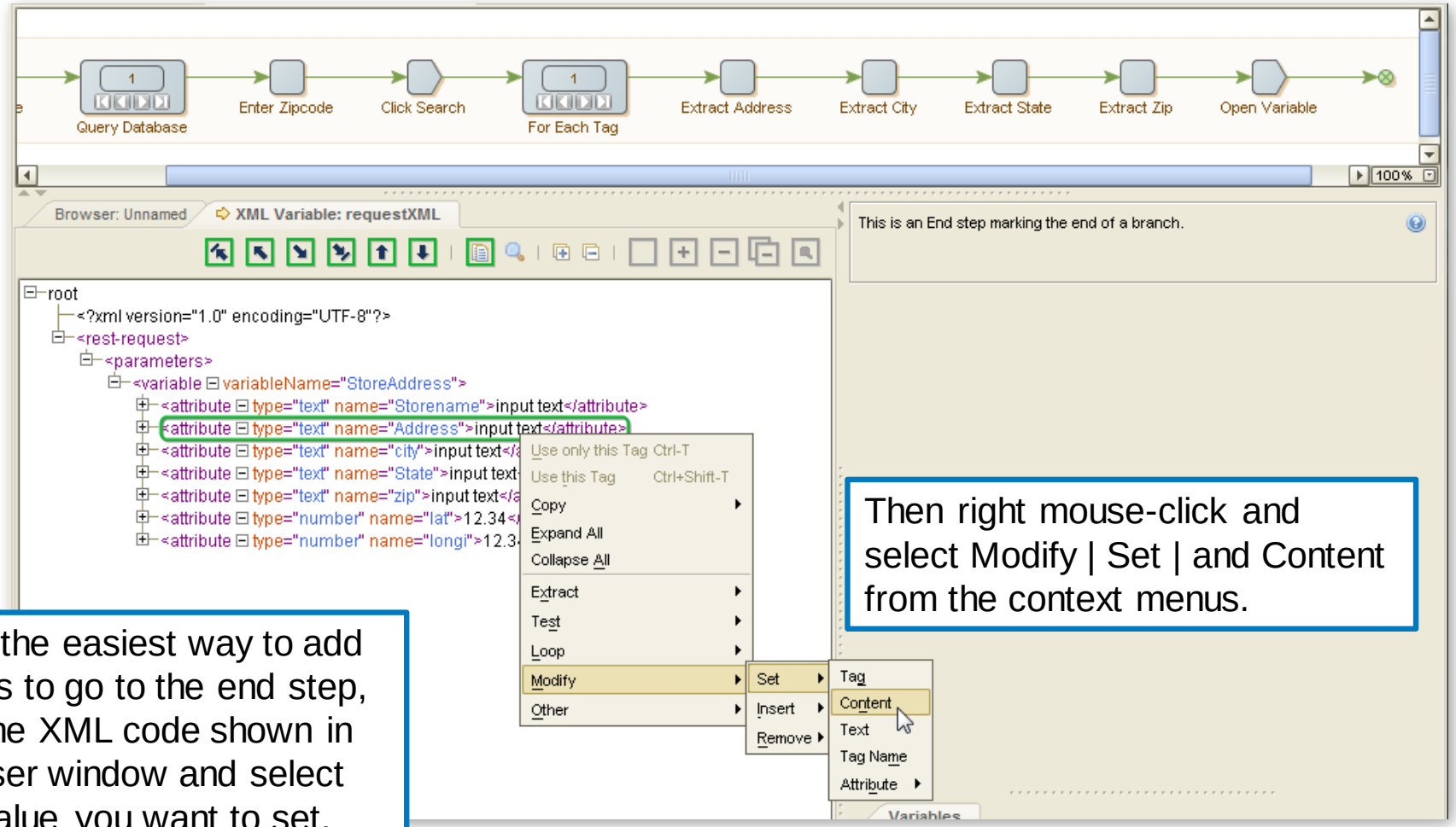
Add Open Variable Action Step

- Next, we will want to open the variable and set the content of the XML code from the data just extracted in the previous Robot steps.
- This replicates what we did when we tested the REST web service earlier in the Management Console by manually entering in the values.



Set Content

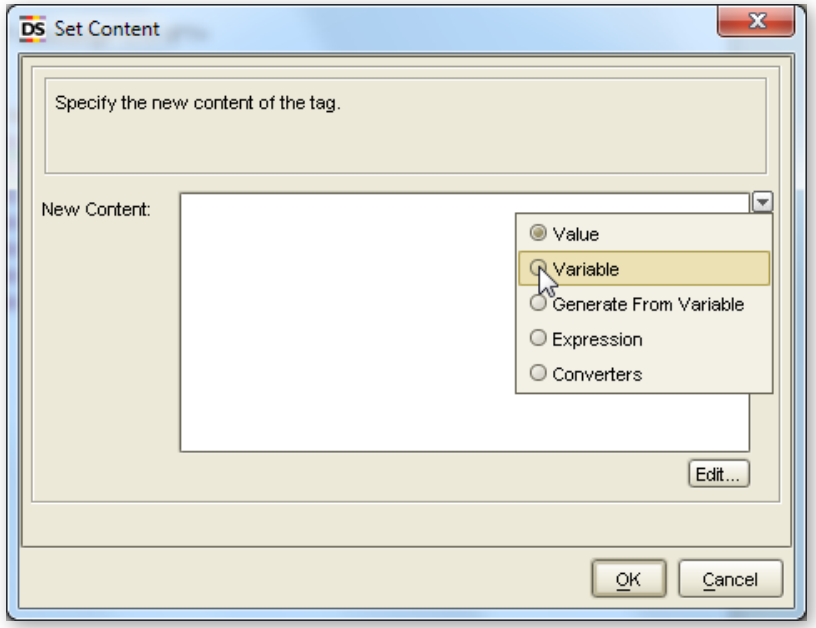
- Then we will add Set Content steps for each of the four values extracted in the earlier steps.



Probably the easiest way to add the step is to go to the end step, expand the XML code shown in the browser window and select the first value you want to set.

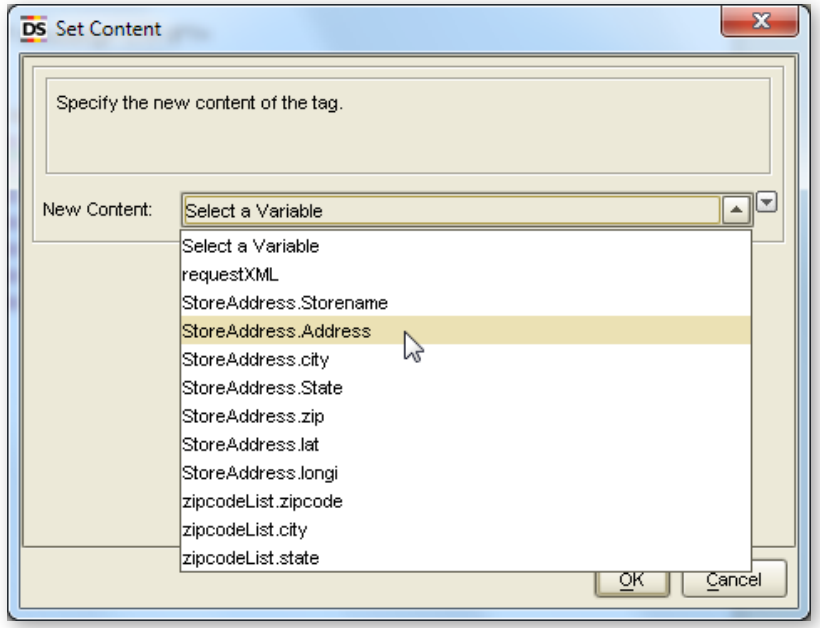
Then right mouse-click and select Modify | Set | and Content from the context menus.

Complete Creating the Set Content Step



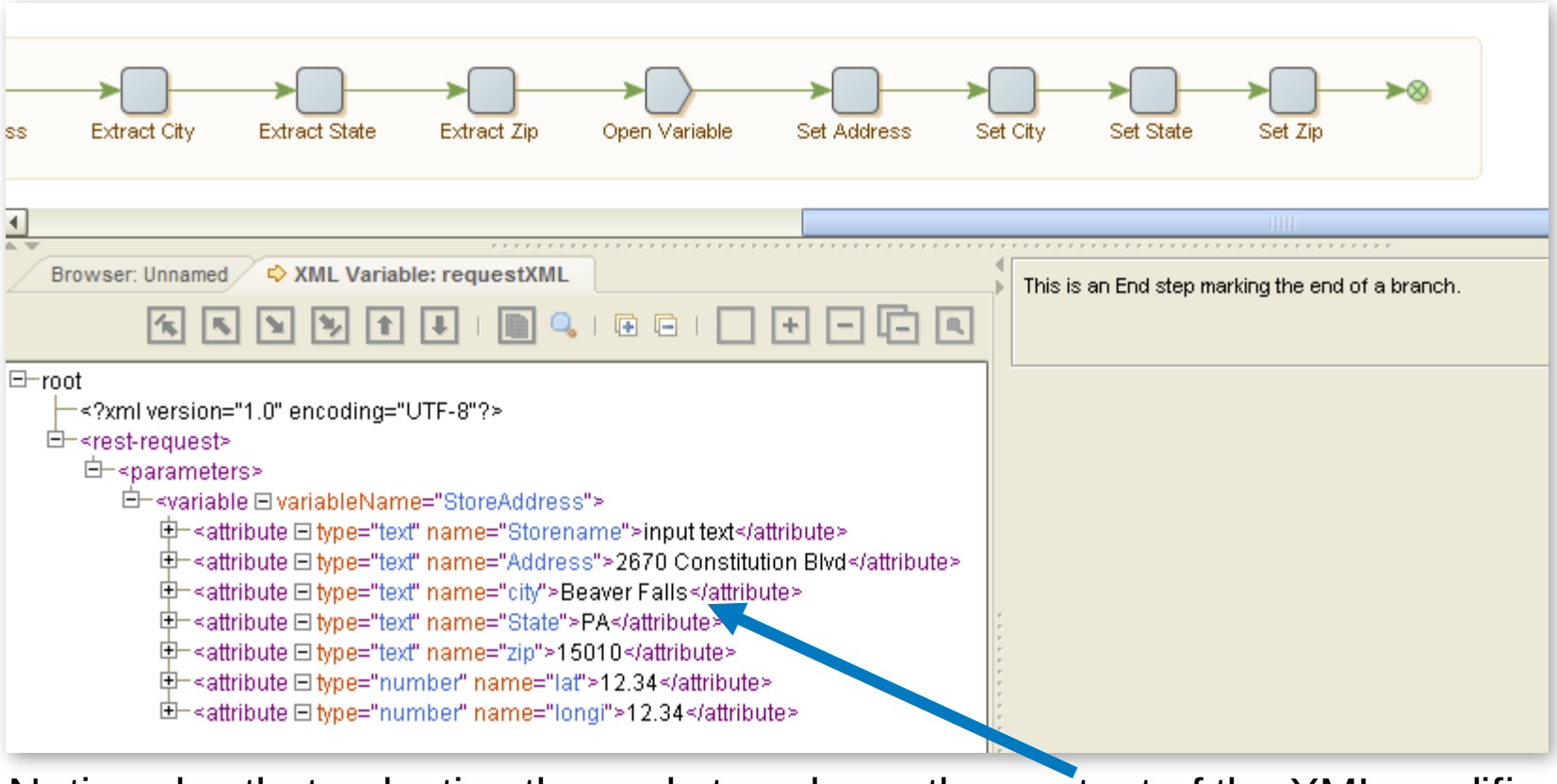
Then specify the new content as a Variable.

…and select which Variable to use as the new content. For this first Set Content step, we're using "StoreAddress.Address."



After Setting Up All Four "Set Content "Steps…

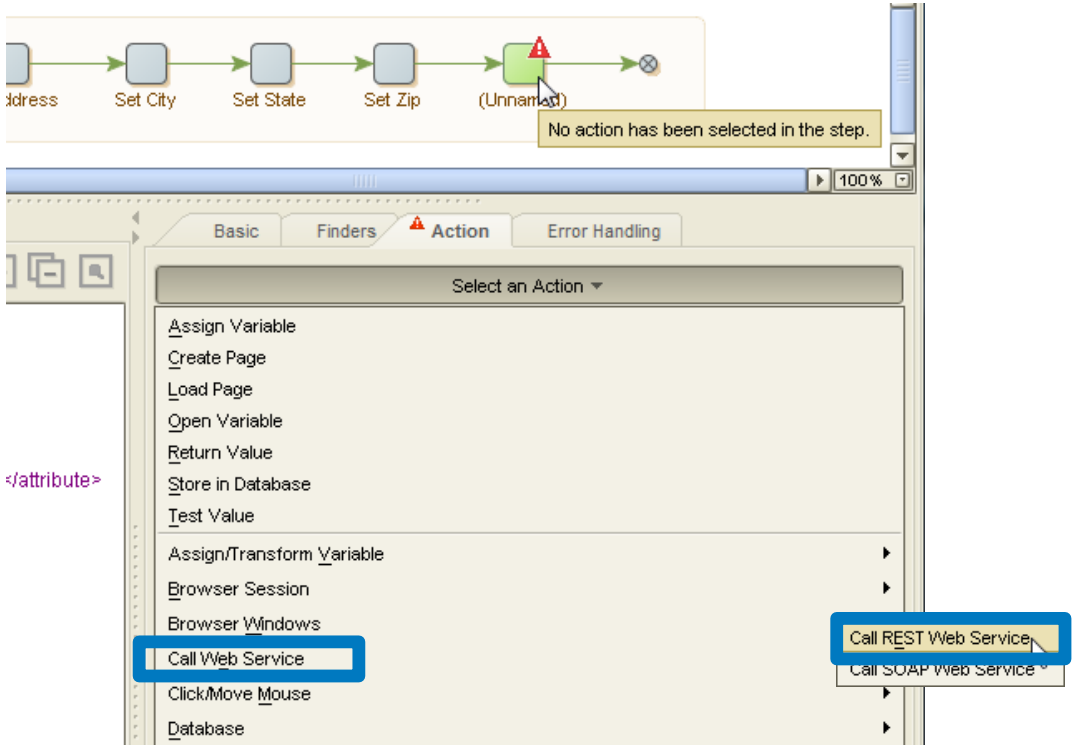
- As you see here, we've set up the other three "Set Content" steps the same way and renamed them (Basic tab) so we can tell what each is doing.



Notice also that selecting the end step shows the content of the XML modified with the data extracted in our earlier Extract Steps.

Then Call REST Web Service

- Now, we need to call our REST Web Service from our earlier Robot to return the corrected address information from our Post Office website along with latitude and longitude. It of course will use the content just provided by the preceding four steps.



Call REST Web Service – What it Does

- The Call REST Web Service action sends a request to a REST web service and returns the web service's response, which may be for instance XML, JSON or HTML. The response is either presented in HTML form as the current page or stored in a variable.
- If the web service returns a fault, the message is not returned by the action. Instead, the action will generate an error which can be handled using the standard error handling mechanisms.

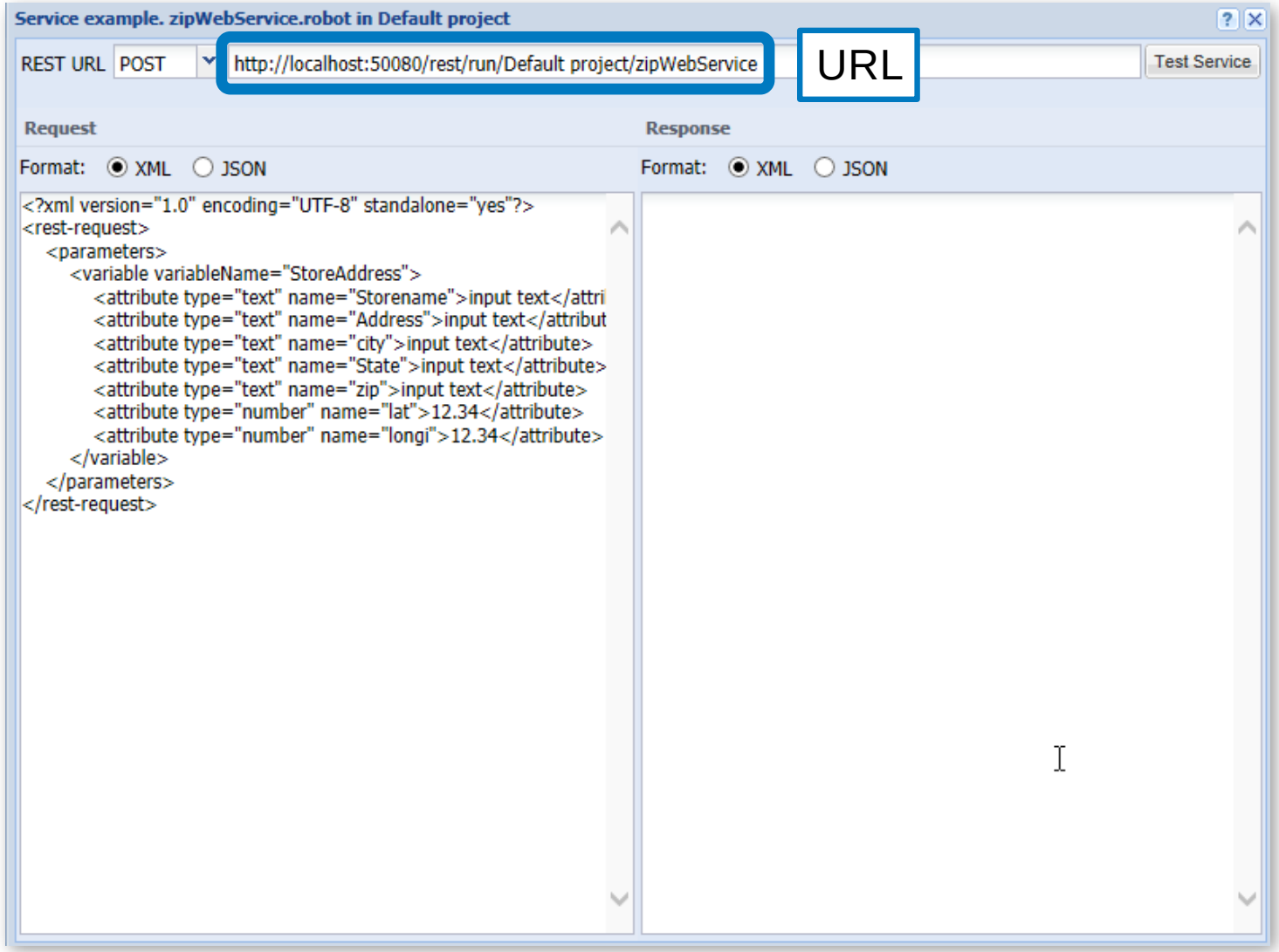
Call REST Web Service Properties

- *URL:* The base URL of the web service, excluding parameters. The URL can be specified in several ways using a *URL Selector*.
- *Request:* Here, you specify the type of request to be made. REST supports four basic operations:
  - *GET:* Used for querying data.
  - *POST:* Used for updating selected parts of data.
  - *PUT:* Used for replacing data.
  - *DELETE:* Used for deleting data. For DELETE requests, you can specify a number of parameters as name/value pairs.

Additional information is available in the online User's Guide.

The URL for Our REST Service

- The URL for our REST Web Service can be discovered in the Management Console by selecting the Robot from the Repository tab and clicking on the REST button.



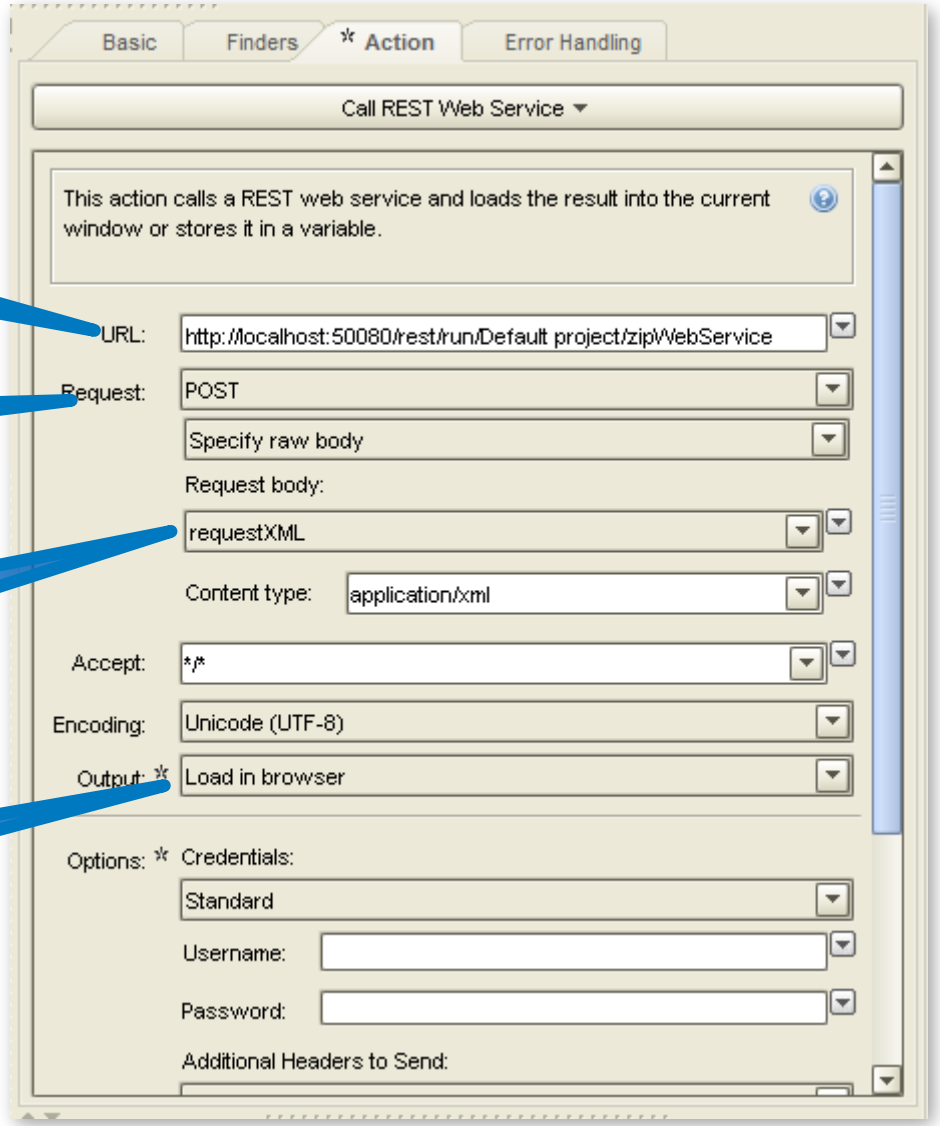
The Properties in Design Studio for Our REST Web Service

The URL

This is a POST request

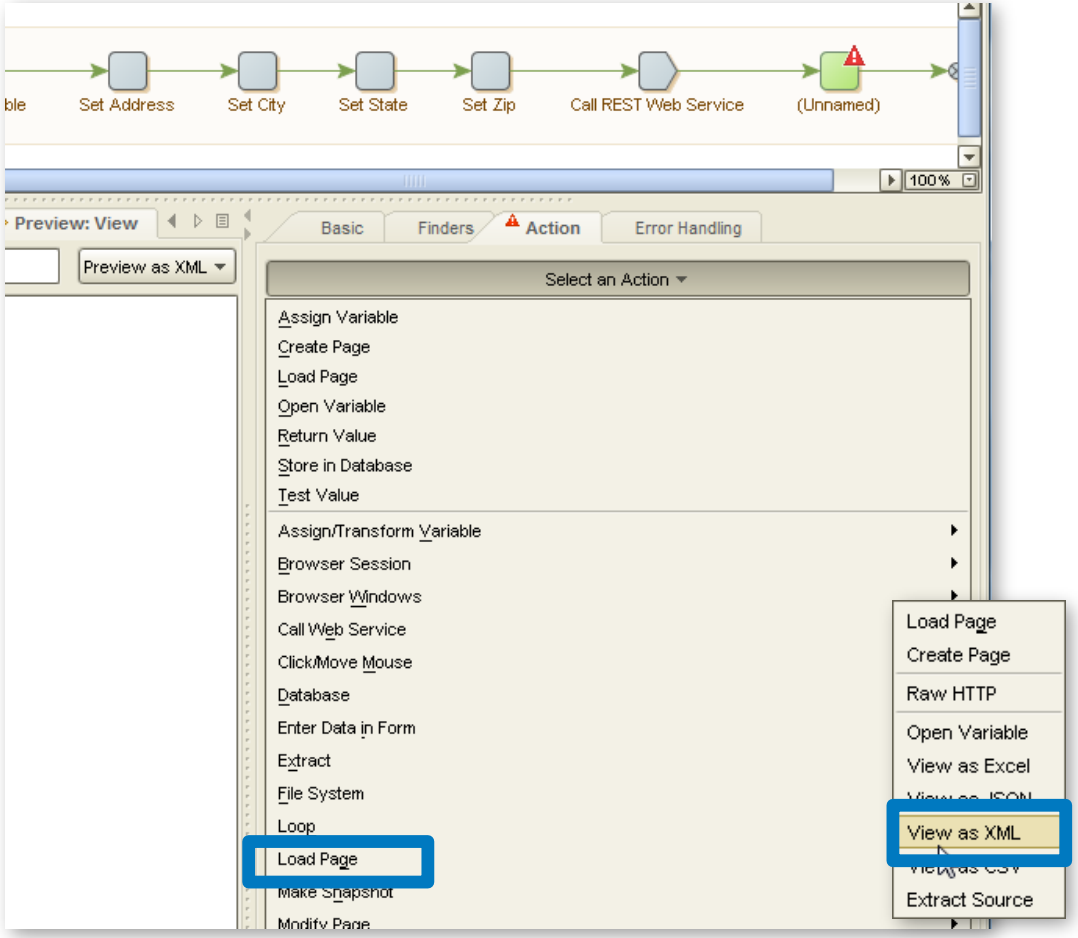
And we are posting to the requestXML Variable we just set up.

The ouput will be loaded into the browser which will then execute the search.



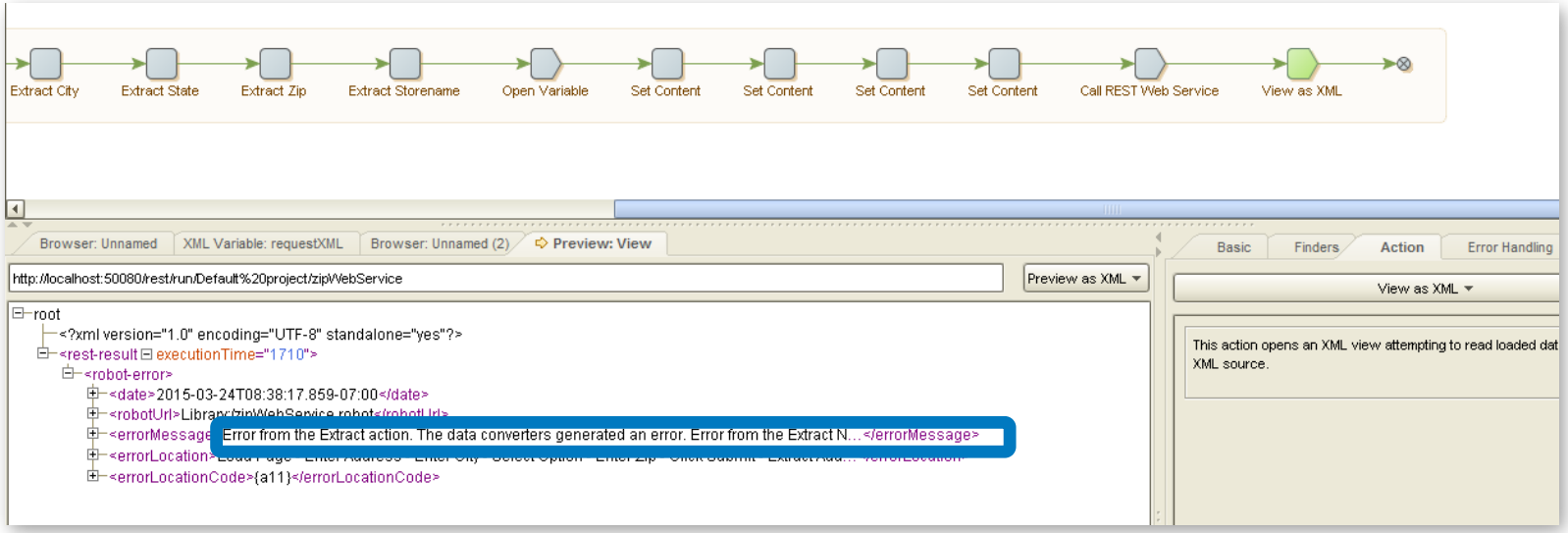
View as XML

- Before we can extract from the data returned by the REST Web Service, we need to load a page that allows us to view that data as XML. So we'll add a new step to do that.

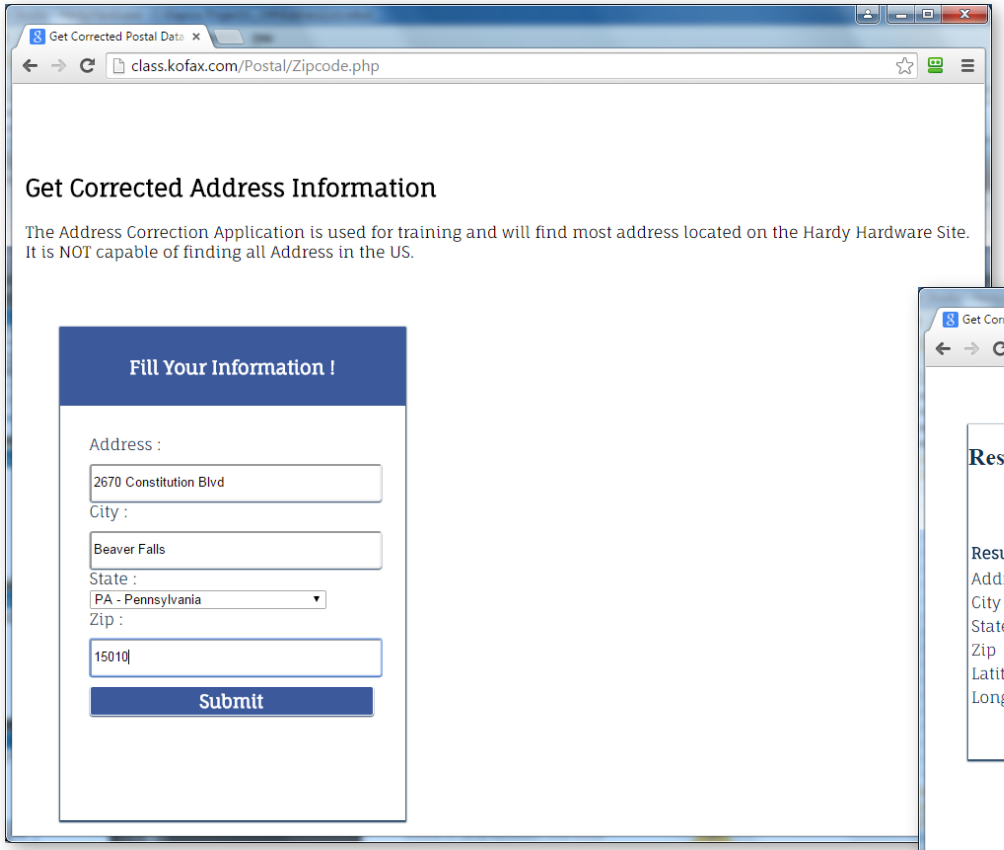


Extract Data from XML

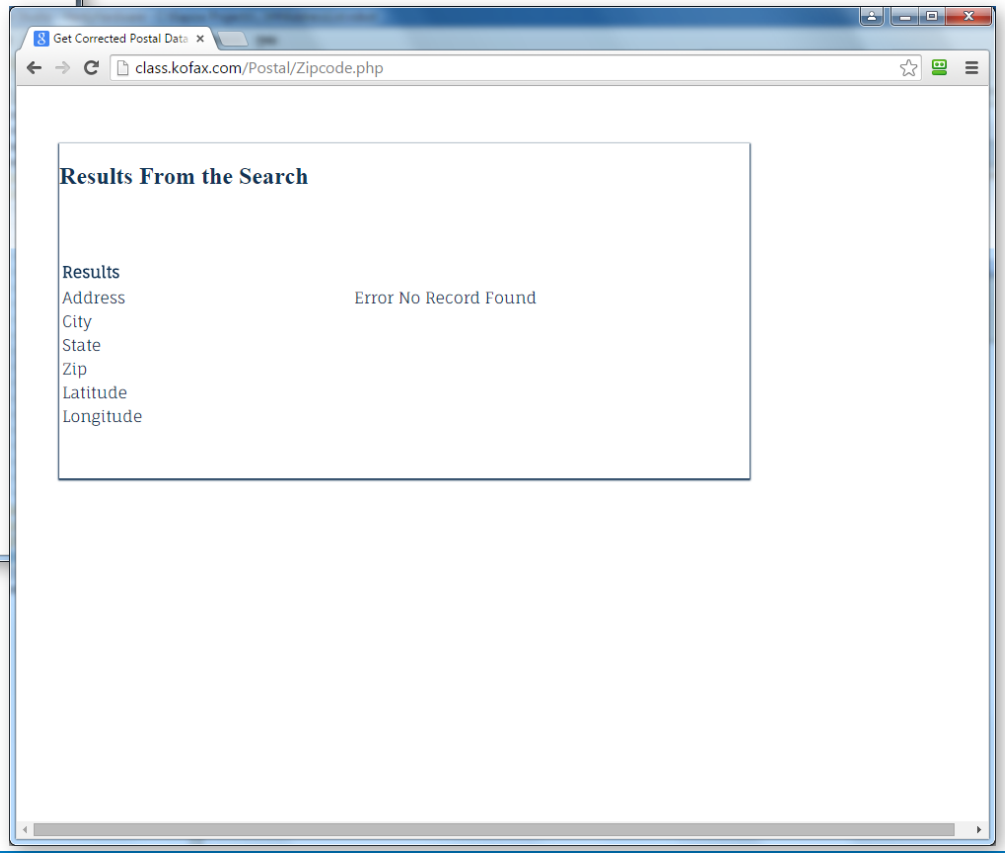
- The next step is to set up Extract Steps to extract the data we want from the XML file displayed in the browser panel.
- This is accomplished in a similar way to extracting a piece of data from a web page.
  - Click on the data you want to extract
  - Right mouse-click and select "Extract" from the context menu.
- BUT…This first record has a problem: There is an error in the XML. This is because no matching data was returned from the Post Office web site.



You Can Verify This As a Human Operator



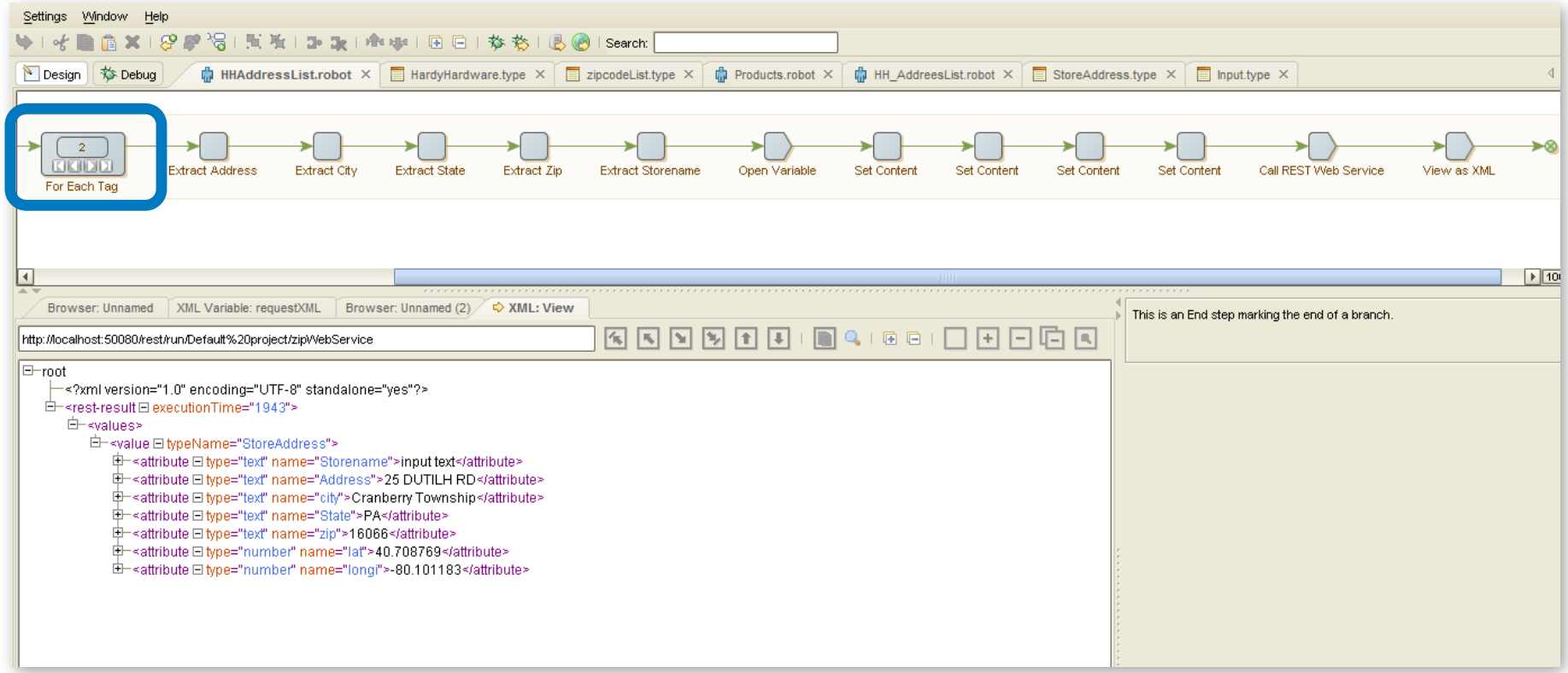
Sometimes you have to troubleshoot robots to determine the cause of the problem.



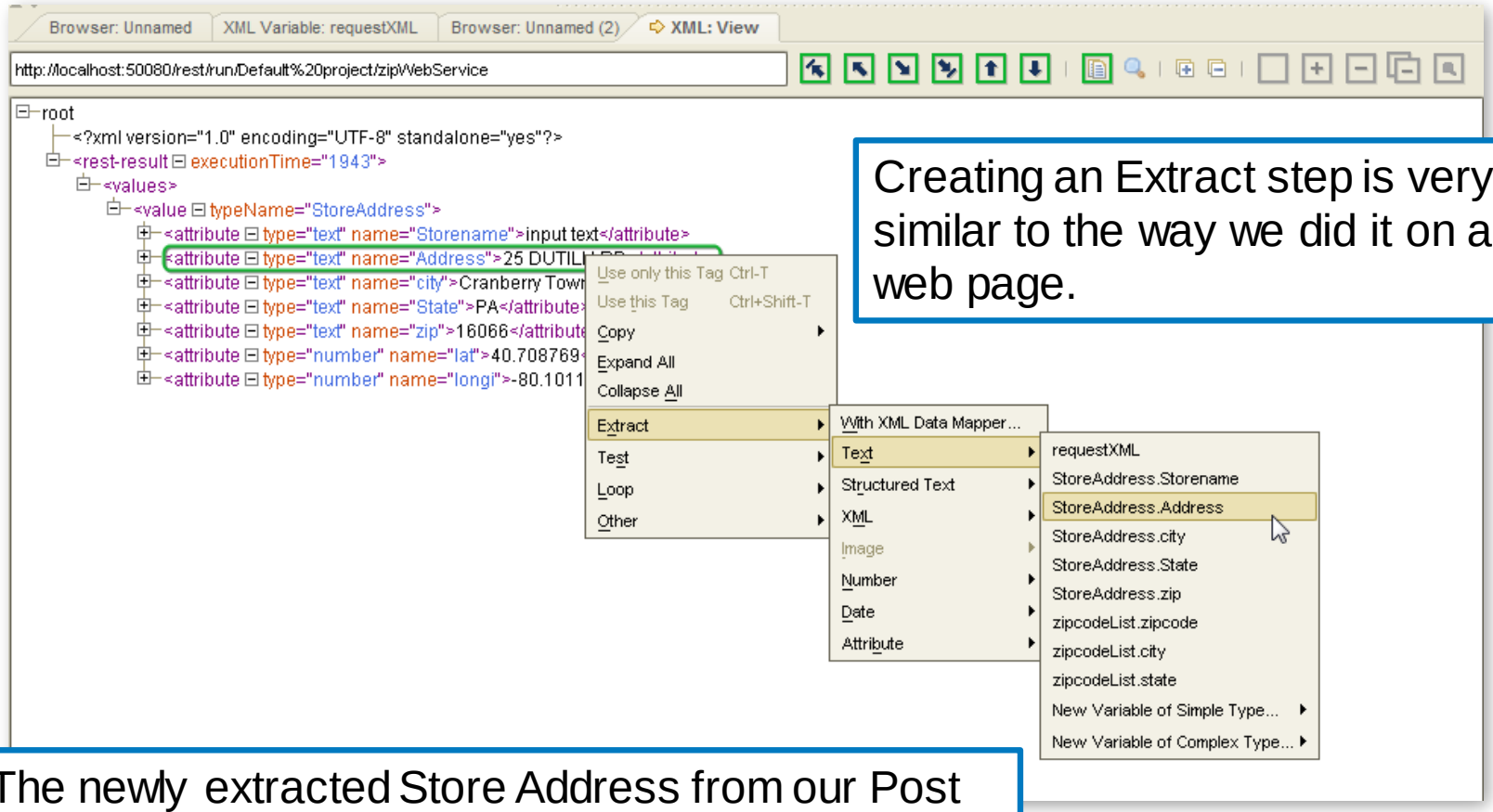
In this case, the Post Office website reveals there was no matching record in the Post Office database.

Let's Test on the Second Record

- By advancing the "For Each Tag" loop, we go to the next record. Clicking on the end step, we see the correct values written to the content of the XML file our REST Web Service will use. Our Robot works after all! We'll build in a way to handle errors where no data is found in a minute. But first, lets' set up extraction…



Extract Corrected City Returned by Web Service

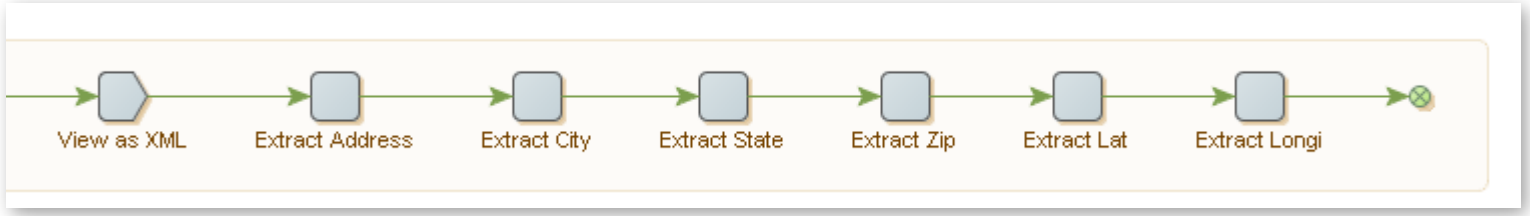


Creating an Extract step is very similar to the way we did it on a web page.

The newly extracted Store Address from our Post Office website will replace the one returned from the HardyHardware website.

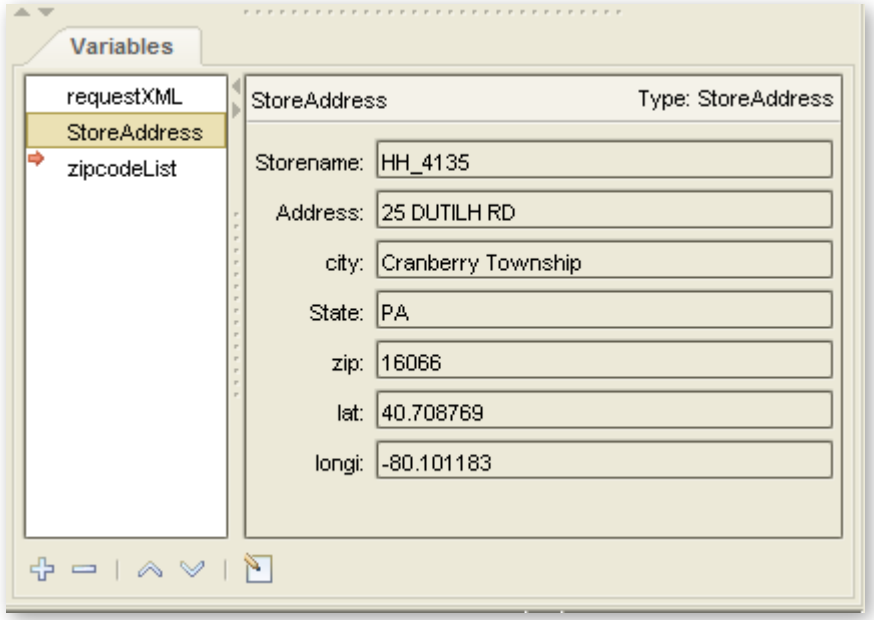
Extract Steps are Added for Other Values

- Extraction is setup to extract City, State, Zip Latitude and Longitude as well. Remember that instead of extracting "Text" for Latitude and Longitude, you will be extracting "Number" instead.



Examining the StoreAddress Variable in the Variables panel shows us that all data was successfully extracted.

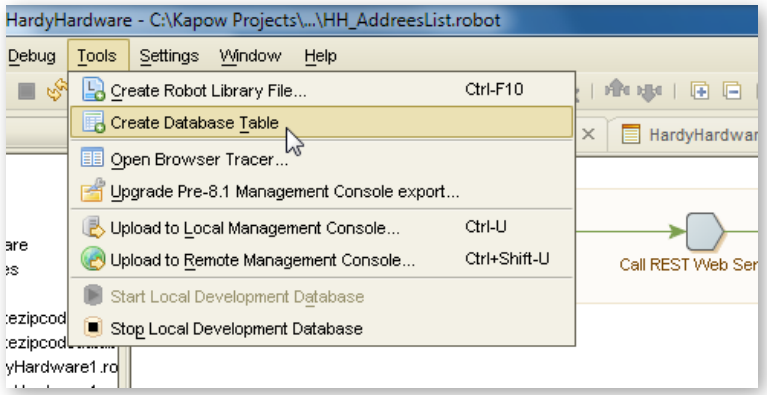
Remember, Storename was extracted in an earlier step and was not part of the work our web service performed.



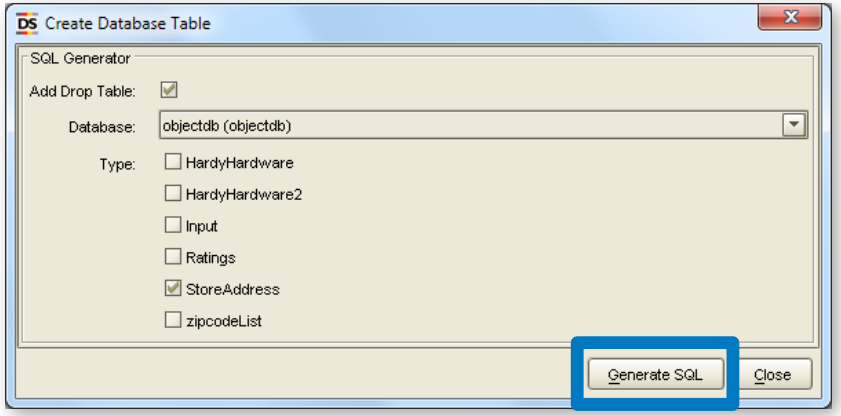
We Want to Write the Results to a Database

- We will set up a "Store in Database" step, but first we have to create the table in our database. Remember how we did this?

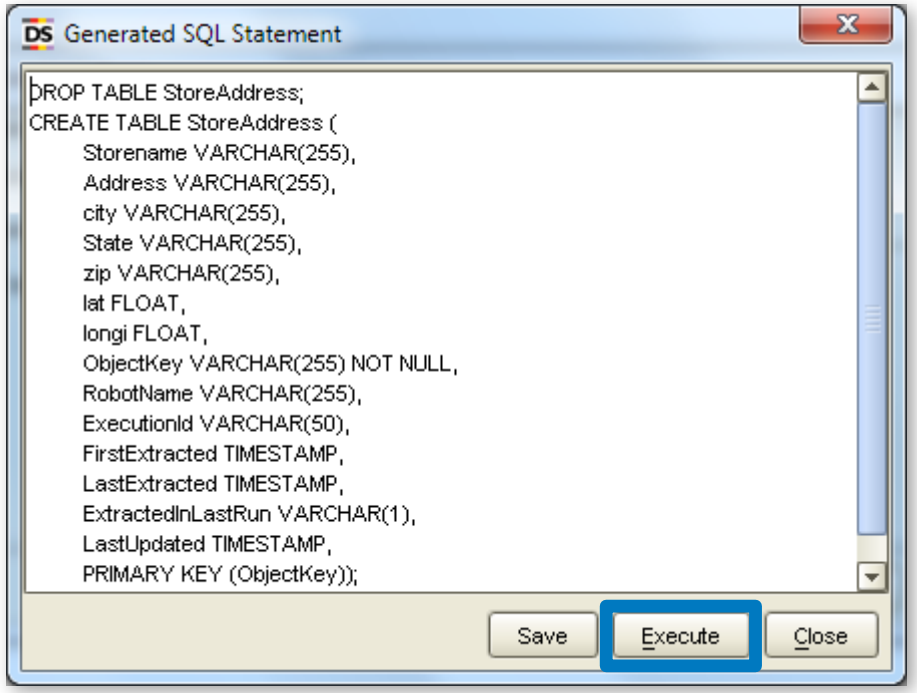
1.



2.

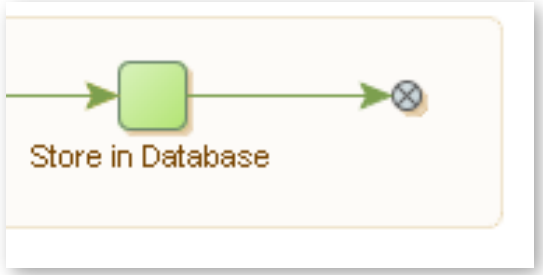


3.

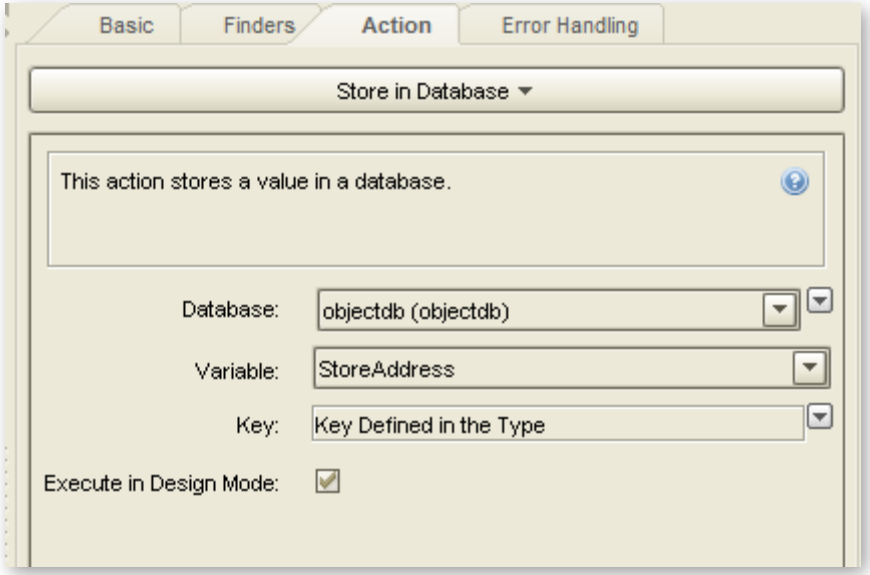


Store in Database

- Finally, we'll add our "Store in Database step.

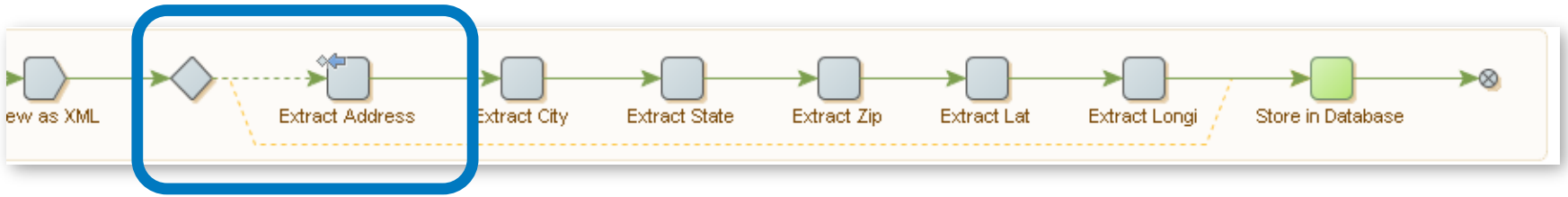


- We are writing to the development database called objectdb. The Variable is StoreAddress which uses the Storename as a primary key. This prevents records from being replicated.



Add Try Step in Case No Data is Found

- BUT WAIT! THERE'S MORE! – Remember that pesky problem that certain store addresses may not be found on our Post Office website?
- We don't want our Robot to stop running with an error. Instead, we want it to bypass any "records not found" and continue running to return records that ARE found.
- A simple "Try" step will accomplish that. We'll add it just before the fist field extracted from the XML (Extract Address). We'll also set Error Handling for the Extract Address step to try next alternative (notice blue arrow pointing back at Try step below).
- Then we'll drag the end step of the branch created to the Store in Database step above. The final result is this:



What's Next?

- After saving your Robot, you could:
  - Run it from Design Studio in Debug mode
  - Upload it to the Management Console, where you could:
    - Run whenever required
    - Schedule to run automatically
    - Create a Kapplet to run it on an ad hoc basis
- Any of the above methods will write data to the database. Whatever solution best meets your need will accomplish them. It's up to you!

Demo & Lab

Calling a Web Service