# Enterprise LI: Using a Spectrum Workflow in MapInfo Pro

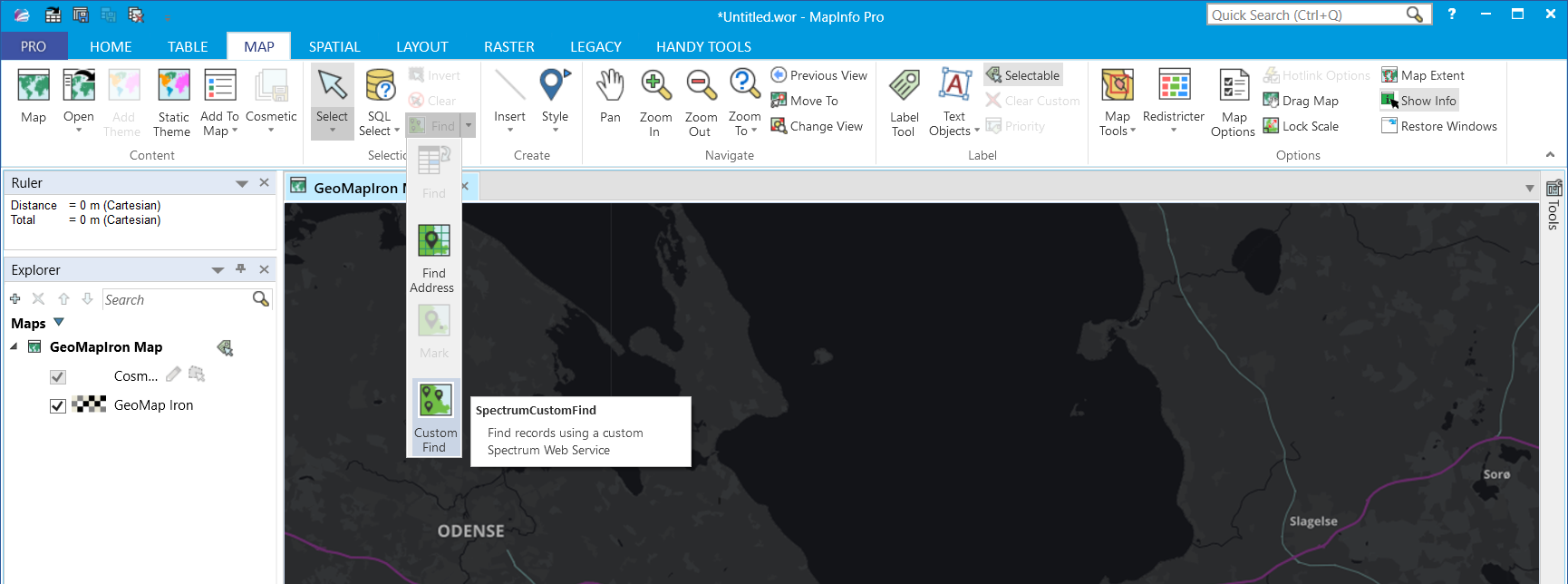
In my latest blog post, I wrote about [creating a workflow in Enterprise Designer](https://community.pitneybowes.com/blogs/peter-horsboll-moller/2018/12/19/creating-a-querying-workflow-in-enterprise-designe). The workflow allowed me to search for records in multiple tables and return the records that met my condition.

In this blog post, we will look at how you can use such a workflow in MapInfo Pro. To do so you need some coding experience, preferable with MapBasic and .NET too. I will share my sample​code so you can use this as a starting point.

I have chosen to split my application into two parts: The interface written with MapBasic and the communication with the Spectrum web service written in C#.

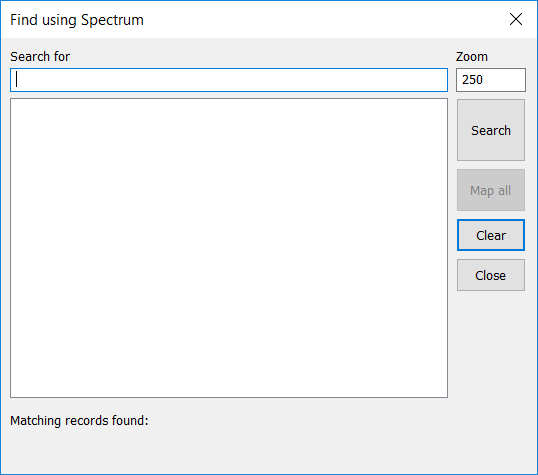
You can do this in multiple ways using other development languages. You can write this entirely using C# or VB.NET keeping the MapBasic part to a bare minimum. You could do it entirely with MapBasic taking advantage of the HTTP functions defined in the HTTPLib define that comes with MapBasic. I do however recommend using a different development language than MapBasic when you need to connect to a web service.

When you run the application, it places the control launching the *Search* dialog in the *Find* dropdown on the *MAP* tab.



## The Search Dialog in MapInfo Pro

To let the user search for specific value, I need to create a user interface for the user to be able to enter a search value. This interface will also be used to present the matching records once they have been returned by the web service. Finally, the interface should let the user select individual records and show their position on the map.



The user enters a value to search for in the *Search For* field and hits the *Search* button. As the query on the server side is using the Like operator, the text entered can contain wildcards such as the percentage (%) and the underscore (\_). If no wildcard is found, the web service will add a % at the end of the search string.

## Communication with the Web Service

When the user has entered a value in the dialog and hit the *Search* button, I use MapBasic to read the entered value, check it and pass the search value onto a .NET method. Before passing it to the C# method, I check if it meet some basic conditions such as holding at least four characters. This can be modified depending on the specific service, the number of and size of data sets you are searching. I call the C# method ***SCFWSDoSearch*** to pass the search value to the web service and return the number of records found.  
  
  Dialog Preserve  
   '\*\*The search value is read from the dialog and set to a property   
   Call DLGFUSCFSetValueToSearchFor(ReadControlValue(CTRL\_TXT\_SEARCH\_FOR\_VALUE))  
   '\*\*Web Service settings are passed to the .NET assembly   
   Call SCFWSInitiate(20, DLGSCFCGetURL(), DLGSCFCGetUserName(), DLGSCFCGetPassword())  
   '\*\*Checking length of the search string   
   If Len(DLGFUSCFGetValueToSearchFor()) Between 1 AND 3 Then   
      Note "Please enter a value with at least 4 characteres!"   
      Alter Control CTRL\_TXT\_SEARCH\_FOR\_VALUE Active   
      Exit Sub   
   End If  
     
   '\*\*Calling the Search Method in the .NET assembly   
   nNumMatchFound = SCFWSDoSearch(DLGFUSCFGetValueToSearchFor())   
...

Using the C# code, I am initializing a HttpWebRequst object and passing the URL and the search value to this request. The response is a (Geo)JSON array that I extract the most used values from into a number of List variables. In my example, I am currently only using the centroid coordinate of the matching records. However, I could also use the MBR of the matching record or the actual spatial object. All these three values are returned by the web service. The web service could be changed to only return the full spatial object and the MapBasic code could be changed to extract the needed values from the full spatial object. There are multiple options here.

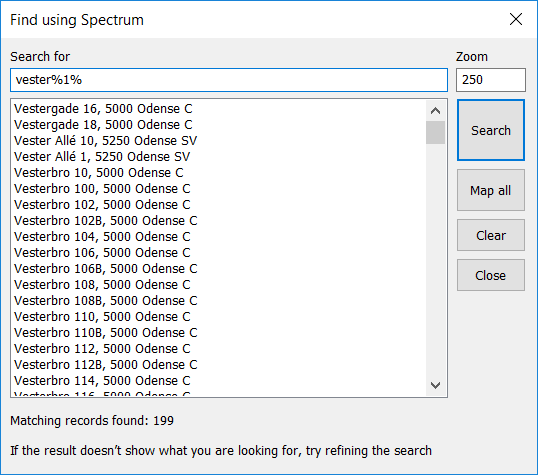
\_latestURL = string.Empty;  
  
//Creating the URL including the search value  
if (valueSearchFor != string.Empty)  
   \_latestURL = string.Format("{0}?Data.SearchValue={1}", \_serverURL, valueSearchFor);  
else  
   return 0;  
  
//Initializing the web request  
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(\_latestURL);  
  
request.Method = "GET";  
request.ContentLength = 0;  
request.ContentType = "text/xml";  
if (\_userName != "")  
   request.Credentials = new NetworkCredential(\_userName, \_passWord);  
  
try  
{  
   //running the web request  
   HttpWebResponse response = (HttpWebResponse)request.GetResponse();  
   string responseValue = string.Empty;  
  
   if (response.StatusCode != HttpStatusCode.OK)  
   {  
      string message = String.Format("Request failed. Received HTTP {0}", response.StatusCode);  
      throw new ApplicationException(message);  
   }  
  
   using (Stream responseStream = response.GetResponseStream())  
   {  
      using (var reader = new StreamReader(responseStream))  
      {  
         //Handling the response  
         string responseBody = reader.ReadToEnd();  
  
         JObject jsonResult = JObject.Parse(responseBody);  
         JArray \_results = jsonResult["Output"].Value<JArray>();  
  
         \_lstDescriptions = new List<string>();  
         \_lstLatitudes = new List<string>();  
         \_lstLongitudes = new List<string>();  
  
         //loading the resonse values into a number of private List variables  
         foreach (dynamic resultItem in \_results)  
         {  
            if ((string)resultItem["Status\_Code"] == "101")  
            {  
               //Ignore this record - might get returned by the web service for no matching records  
            }  
            else if ((string)resultItem["Status\_Code"] == "NoMatchingRecordsFound")  
            {  
                //Ignore this record - might get returned by the web service for no matching records for the individual query sub flows  
            }  
            else  
            {   
               \_lstDescriptions.Add((string)resultItem["VALUE"]);  
               \_lstLatitudes.Add((string)resultItem["Y"]);  
               \_lstLongitudes.Add((string)resultItem["X"]);  
            }  
         }  
         return \_lstDescriptions.Count;  
      }  
   }  
}  
catch (WebException e)  
{  
   if (e.Status == WebExceptionStatus.ProtocolError)  
   {  
      MessageBox.Show(String.Format("DoSearch: Status Description : {0} {1}", ((HttpWebResponse)e.Response).StatusDescription, e.Message));  
   }  
}  
return 0;

You could use this to call different web services as long as the web service returns a similar structure. You only have to specify the correct URL and make sure the name of the parameter you are passing to the web service is *Data.SearchValue*. That can be modified in the code so that the name of this parameter is passed to the .NET method from the MapBasic application and so no longer is a hard-coded string.

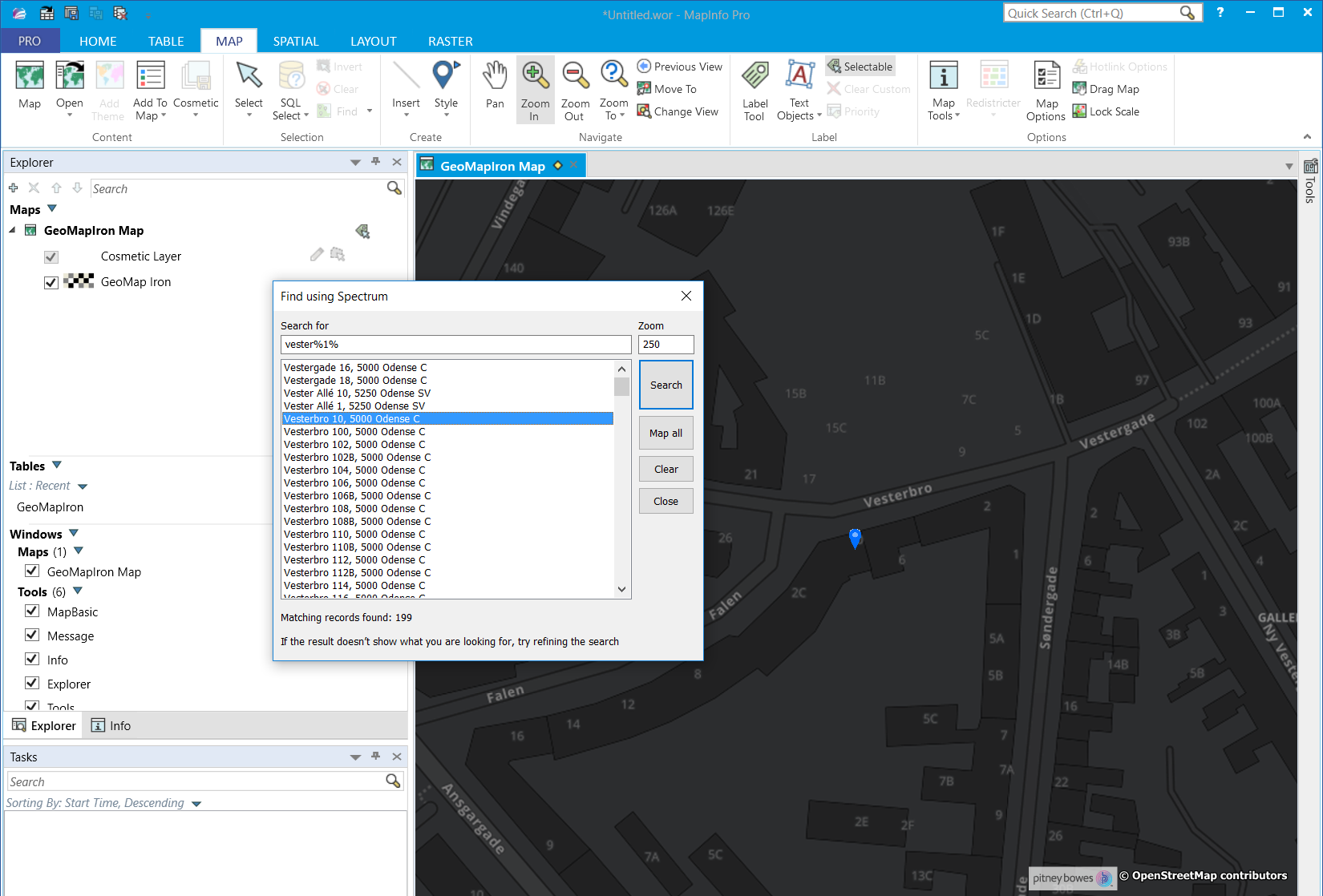
## Showing the Matching Records

Once you have found some records you want to present these to the user. The actual values to show is determined by the Web Service and are available as the Description value. A lot of the code below is validating the search value and the number of return values.

The method ***SCFWSGetResultDescriptions***copies the content of the Descriptions List in C# to my MapBasic array so that I can publish my ListBox control with the array. Publishing the ListBox control is done in the last line of the example below.

...  
   '\*\*Checking the number of found records   
   If nNumMatchFound = -1 Then   
      Alter Control CTRL\_LBL\_NUM\_RECORDS\_FOUND Title "Matching records found: "   
   ElseIf nNumMatchFound = -2 Then   
      Alter Control CTRL\_LBL\_NUM\_RECORDS\_FOUND Title "Matching records found: 0"   
   Else   
      Alter Control CTRL\_LBL\_NUM\_RECORDS\_FOUND Title "Matching records found: " & FormatNumber$(nNumMatchFound)   
   End If  
  
   If nNumMatchFound >= DLGFUSCFGetMaximumMatchNumber() OR nNumMatchFound = -1 Then   
      Alter Control CTRL\_LBL\_REFINE\_INFO Title "The search returned more than " & FormatNumber$(DLGFUSCFGetMaximumMatchNumber())   
                                              & " records. Try refining the search"   
   ElseIf nNumMatchFound = -2 Then   
      Alter Control CTRL\_LBL\_REFINE\_INFO Title "The search returned 0 records. Try refining the search"   
   Else   
      Alter Control CTRL\_LBL\_REFINE\_INFO Title "If the result doesn’t show what you are looking for, try refining the search"   
   End If  
  
   If nNumMatchFound > DLGFUSCFGetMaximumMatchNumber() OR nNumMatchFound <= 0 Then   
      Redim marrResults(0)   
      Alter Control CTRL\_TXT\_SEARCH\_FOR\_VALUE Active   
   Else   
      Redim marrResults(nNumMatchFound)   
      '\*\*Retrieving the records found from the .NET assembly   
      If SCFWSGetResultDescriptions(marrResults) = nNumMatchFound Then   
         '\*\*All matching records returned  
      End If   
   End If  
  
   '\*\*Publishing the list with the matching records   
   Alter Control CTRL\_LST\_RESULTS Title From Variable marrResults  
  


Now I have a list of descriptions in my ListBox control. When the user selects one of these, I can query the X and Y coordinates of that specific record using two .NET methods: **SCFWSGetResultX** and **SCFWSGetResultX**. Then the map can be zoomed and panned to the location of the selected record.



## A few things to consider

This is still a Proof of Concept even though it can be used in production. You need to consider a few things if you want to use this tool.  
  
**Coordinate system**: Currently the coordinate system of the application is either hard-coded to UTM Zone 32 ETRS89, a Danish coordinate system or using the coordinate system of a dedicated table called ApplicationCoordSys . This means that the service also needs to return the values, the GeoJSON elements using this coordinate system. You can get around this in a number way if you want to make it more flexible.

* You can specify that the service always is using Lat/Long WGS84 as this would cover the World.
* You could specify the coordinate system using an EPSG code when calling the service and then have the service do the coordinate transformation before passing back the coordinates in the GeoJSON objects.
* Finally, the simple solution: You can just change the hardcoded coordinate system in the MapBasic application or the coordinate system of the table mentioned above to match the coordinate system of your datasets.

**Web Service Parameters**: As long as you are only using a single parameter the easy solution is to make sure your Spectrum service is using the same parameter name as I mentioned above. If you start using services with multiple parameters, you will have to adopt these in the application too. One option is to pass two arrays of values to the search method. One array of parameter names and another array of the search values. This would give you a very flexible solution.

**A dockable Panel instead of a modal dialog**: A search tool like this should be placed in a dockable panel so that it can be open all the time. Look at the source code for the NamedViews tool that comes with MapBasic. This will give you some ideas on how to build a custom panel for the dockable search window. This would be the choice if you want to use .NET more in your development.

**Accessing the GeoJSON data**: If you want to access the actual spatial object stored in the GeoJSON response, look at the MapInfo Data Access Library, MDAL, that was added to MapInfo Pro 17.0. A .NET library allows you read a GeoJSON structure, convert it to a MapInfo spatial object and save that to a MapInfo table. This will also port more of the source code from MapBasic to .NET.

## Getting up and running

I have shared the source code for the project on GitHub. You can get it here: https://github.com/PeterHorsbollMoller/mbSpectrumCustomFind

It contains the MapBasic code, the .NET code and the Spectrum dataflow.

1. You need to create a dataflow on your Spectrum instance as I described in my earlier blogpost. You can use one of the dataflows from the GitHub project but make sure to change the reference to the named resources in the dataflow.
2. You probably need to change the coordinate system using by the MapBasic application to match the coordinate system of your Spectrum Web Service. Open the table ApplicationCoordsys, drop the map of the table and make the table mappable again using the desired coordinate system.
3. Run the MapBasic application and check the configuration via the context menu of the application in the Tools window. Change the settings to match your setup, see image below.
4. Now access the Custom Find method in the Find dropdown on the MAP tab.

