Store Locator Tutorial

INTRODUCTION

This tutorial is designed to assist you in creating a Store Locator type application for your own website. A general knowledge of Databases and the programming language of your choice is required.

The sample application and database provided here will run out of the box on any Microsoft IIS environment.

REQUIREMENTS

For your own Store Locator Application, you will need the following:

1. Any U.S. ZIP Code Database provided by Zip-Codes.com
2. Database of stores with Name, Address, and other pertinent information generated by you.

LANGUAGE

This tutorial will provide specific code sample in the VBScript language only. The focus is not on specific code, but instead to offer guidance and logic necessary to create a Store Locator application in your own language. We cannot provide samples in every language available.

SAMPLES

This tutorial comes with two sample files:

1. Sample Access Database (StoreLocator.MDB)
2. Sample ASP Script (StoreLocator.asp)

YOUR APPLICATION

Your Store Locator application can be easily developed by following these steps:

1. Setting up Your Database
2. Programming: Accept the users ZIP Code

3. Programming: Get the users ZIP Code Information
4. Programming: Select Nearby Stores
5. Programming: Display Store Locations

SETTING UP YOUR DATABASE

The first step in creating a Store Locator is to have a list of every ZIP Code and it's Latitude/Longitude coordinates. Zip-Codes.com provides 3 different databases that contain the required information for purchase and can be found here:

http://www.zip-codes.com/zip-code-database.asp

A sample of the data would look like this:

Table: ZipCodes					
ZipCode	**City**	**State**	**County**	**Latitude**	**Longitude**
10006	NEW YORK	NY	NEW YORK	40.708135	-74.013363
10006	NEW YORK	NY	NEW YORK	40.708135	-74.013363
10006	NEW YORK	NY	NEW YORK	40.708135	-74.013363
10006	NEW YORK	NY	NEW YORK	40.708135	-74.013363
32504	PENSACOLA	FL	ESCAMBIA	30.488521	-87.187298
35046	CLANTON	AL	CHILTON	32.840013	-86.624704
35048	CLAY	AL	JEFFERSON	33.735224	-86.569042
75966	NEWTON	TX	NEWTON	30.842072	-93.759271
75968	PINELAND	TX	SABINE	31.247324	-93.977915
90210	BEVERLY HILLS	CA	LOS ANGELES	34.096629	-118.412426
90212	BEVERLY HILLS	CA	LOS ANGELES	34.061258	-118.401293

You can use the Access Database provided by Zip-Codes.com, or import the dataset into your own SQL Server, MySQL, or Oracle Database. For the purposes of this tutorial, we assume the table name to be "ZipCodes"

The second step is to get your database setup with a list of stores and the information for each store. For each Store, you need to store the following information:

1. Store Name
2. Address
3. City
4. State
5. ZIP Code

Any other fields you deem necessary can also be stored, but these are the required values.

A sample of what your database might look like:

Table: Stores						
stNumber	**stName**	**stAddress**	**stCity**	**stState**	**stZIP**	**stPhone**
0026	My Widget Store #0026	123 Someplace St	Clanton	AL	35046	123-456-7890
0048	My Widget Store #0048	385 Some St	New York	NY	10006	123-456-7890
0092	My Widget Store #0092	504 Ninth Ave	Pensacola	FL	32504	123-456-7890
0010	My Widget Store #0010	3524 Creighton Rd	Newton	TX	75966	123-456-7890
0165	My Widget Store #0165	256 Sunset Blvd	Beverly Hills	CA	90212	123-456-7890

For the purposes of this tutorial, we assume the table name to be “Stores”

Once you have the database setup and the data input, you can begin the programming.

PROGRAMMING: ACCEPT THE USERS ZIP CODE

The system should allow the user to enter their ZIP Code in for the search and possibly a maximum distance for miles. For a web application, this can be done via an HTML form POST.

Example:

```
<form name="Form" method="post">
<p style="font-size:11px; font-family:Verdana, Arial, Helvetica, sans-serif;">
  Search for Stores within:
  <input name="fMiles" type="text" id="fMiles" value="<%=fMiles%>" size="5"
maxlength="5">
  miles of zip code
  <input name="fZip" type="text" id="fZip" value="<%=fZip%>" size="5"
maxlength="5">
  .
  <input type="submit" name="Submit" value="Search">
</p>
</form>
```

Next, you must set the application to collect the users values before we can do anything with it.

Example:

```
'--> Collect the fMiles value. If a numeric value is NOT passed, use default
if (request("fMiles") <> "" and isnumeric(request("fMiles"))) then fMiles =
clng(request("fMiles")) else fMiles = 25
```

```
'--> Collect the fZip value. This is the users 'home' ZIP Code.
fZip = request("fZip")
```

PROGRAMMING: GET THE USERS ZIP CODE INFORMATION

After the user has entered in their ZIP Code, we need to look up the data in relation to that ZIP Code. Specifically, we need the Latitude and Longitude values.

```
set chk = dbConn.execute("SELECT Latitude, Longitude FROM ZipCodes WHERE
ZipCode = '" & replace(fZip, "'", "''") & "'")
  if chk.EOF then
    '--> ZIP Code does not exist, print error message
    %><p style="color:#FF0000">Error: ZIP Code not found.</p><%
  Else
    '--> Set Lat/Long values to variables
    baseLatitude = cdbl(chk("Latitude"))
    baseLongitude = cdbl(chk("Longitude"))
  end if
set chk = nothing
```

PROGRAMMING: SELECT NEARBY STORES

With this programming in place, you now have the information you need to find the stores that are within the given range of the users location.

SQL does not have any easy to use functions for calculating distance. It's also impractical to select all stores and calculate the distance. To help alleviate this problem, we can tackle it in a different manner.

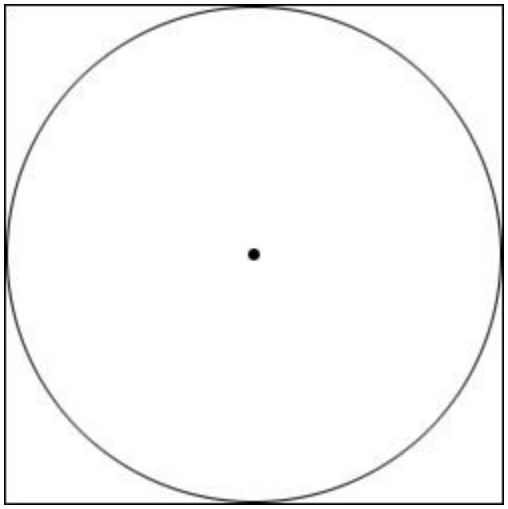

Figure A

In Figure A, the dot represents the location of your client. The circle represents the area that stores should be located in for the results. The square represents the method we can use to narrow down the results.

In essence, we want to pull all stores from the database that fall within the square radius. Then we can run a distance function to determine the stores actual distance from the user and see if it falls within the specified range.

Calculating the square radius is easy. We already have the users Latitude and Longitude. Getting the range can be handled by two functions.

Example:

```
function RadiusAddonLat(varLatitude, varMiles)
  RadiusAddonLat = varMiles / ((6076 / 5280) * 60) + varMiles / 1000
end function


function RadiusAddonLon(varLatitude, varMiles)
  RadiusAddonLon = varMiles / (((cos(cdbl(varLatitude * 3.141592653589 / 180))
* 6076) / 5280) * 60) + varMiles / 1000
end function
```

By using these functions, we can determine the value to add or subtract to the base Latitude and Longitude for the square radius. Then we can use that in our SQL Statement to select all stores.

Example:

```
'--> Get the Lat/Lon Add/Subtract values
LatValue = RadiusAddonLat(baseLatitude, fMiles)
```

```
LonValue = RadiusAddonLon(baseLatitude, fMiles)

Create SQL Statement to pull out stores from the database
SELECT * FROM Stores INNER JOIN ZIPCodes ON Stores.stZIP = ZIPCodes.ZipCode"
WHERE City = CityAliasName AND (Latitude <= (baseLatitude + LatValue) OR
Latitude >= (baseLatitude - LatValue)) AND (Longitude <= (baseLongitude +
LonValue) OR Longitude >= (baseLongitude - LonValue)) ORDER BY stName ASC
```

Use this SQL Query to select the records.

Example:

```
set getRecords = dbConn.execute(strSQL)
```

PROGRAMMING: DISPLAY STORE LOCATIONS

When executed properly, the SQL Statement above will pull all stores from your database that are within range of the user (with a few that are a bit further away).

Next, you have to loop through the records and determine if they should be displayed. For each record, you can use the distance calculation function to determine it's actual distance from the users point.

Distance Function:

```
' Distance Calculator
'=====================================================================
' This function calculates the distance between two latitude/logitude
' coordinates. Disance can be returned as Nautical Miles, Kilometers,
' or Miles (default).
'
'
' This function was designed for VBScript
'
' Accepts:
'      Lat1 = Latitude of point one (decimal, required)
'      Lon1 = Longitude of point one (decimal, required)
'      Lat2 = Latitude of point two (decimal, required)
'      Lon2 = Longitude of point two (decimal, required)
'      UnitFlag = Non required character K, N, or M
'             where:
'                    K = Kilometers
'                    N = Nautical Miles
'                    M = Miles [default]
'
' Provided by: http://www.zip-codes.com
'=====================================================================
Public Function DistanceCalc(Lat1, Lon1, Lat2, Lon2, UnitFlag)
    Dim LatRad1, LonRad1, LatRad2, LonRad2, LonRadDif, RadDist, X, PI, DistMI
    PI = 3.141592654
```

```
    If IsNull(Lat1) Then
        Exit Function
    End If
    If Lat1 = 0 Or Lon1 = 0 Or Lat2 = 0 Or Lon2 = 0 Then
        DistanceCalc = Null
        Exit Function
    ElseIf Lat1 = Lat2 And Lon1 = Lon2 Then
        DistanceCalc = 0
        Exit Function
    End If
    LatRad1 = Lat1 * PI / 180
    LonRad1 = Lon1 * PI / 180
    LatRad2 = Lat2 * PI / 180
    LonRad2 = Lon2 * PI / 180
    LonRadDif = Abs(LonRad1 - LonRad2)
    X = Sin(LatRad1) * Sin(LatRad2) + Cos(LatRad1) * Cos(LatRad2) *
Cos(LonRadDif)
    RadDist = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
    DistMI = RadDist * 3958.754
    If UCase("" & UnitFlag) = "K" Then
            DistanceCalc = DistMI * 1.609344
      ElseIf UCase("" & UnitFlag) = "N" Then
            DistanceCalc = DistMI * 0.8684
      Else
            DistanceCalc = DistMI
      End If
End Function
```

Looping through each record, we set the stores data to variables, and calculate the distance. If the actual distance is within the users selected distance, then we display the record. Otherwise, we do not.

Example:

```
<%
'--> Create Database Recordset and loop through the records
set getRecords = dbConn.execute(strSQL)
  while NOT getRecords.EOF
    '--> Set all store values to variables
    stID = getRecords("stID")
    stName = getRecords("stName")
    stAddress = getRecords("stAddress")
    stCity = getRecords("stCity")
    stState = getRecords("stState")
    stZip = getRecords("stZIP")
    stPhone = getRecords("stPhone")
    stFax = getRecords("stFax")
    stEmail = getRecords("stEmail")
    sLatitude = cdbl(getRecords("Latitude"))
    sLongitude = cdbl(getRecords("Longitude"))

    '--> Use this function to calculate the actual distance for this store
    sDistance = cdbl(FormatNumber(DistanceCalc(baseLatitude, baseLongitude,
sLatitude, sLongitude, "M"), 2, -2, -2, -2))

    '--> Show this record if it's distance is less than what the user entered
    if sDistance <= fMiles then
```

```
%>
  <tr>
    <td><%=sDistance%> Miles</td>
    <td><%=stName%></td>
    <td><%=stAddress%><br><%=stCity%>, <%=stState%> <%=stZip%></td>
    <td><%=stPhone%></td>
    <td><%=stFax%></td>
    <td><a href="mailto:<%=stEmail%>"><%=stEmail%></a></td>
  </tr>
<%
    end if
  getRecords.movenext()
  wend
'--> Close recordset connection
set getRecords = nothing
%>
```

Your Store Locator Application is now complete.