using System;

using System.Windows;

using System.Windows.Media;

using Microsoft.Phone.Controls;

using Microsoft.Phone.Controls.Maps;

using Microsoft.Phone.Controls.Maps.Platform;

using System.Device.Location;

using System.Windows.Shapes;

using findroute.geocodeservice;

using System.Windows.Media.Imaging;

namespace findroute

{

public partial class MainPage : PhoneApplicationPage

{

Location location = new Location();

// Constructor

public MainPage()

{

InitializeComponent();

map1.LogoVisibility = Visibility.Collapsed;

map1.CopyrightVisibility = Visibility.Collapsed;

map1.Mode = new AerialMode();

}

//This method accepts a geocode query string as well as a ‘waypoint index’, which will be used to track each asynchronous geocode request.

private void Geocode(string strAddress, int waypointIndex)

{

// Create the service variable and set the callback method using the GeocodeCompleted property.

findroute.geocodeservice.GeocodeServiceClient geocodeService = new findroute.geocodeservice.GeocodeServiceClient("BasicHttpBinding\_IGeocodeService");

geocodeService.GeocodeCompleted += new EventHandler<findroute.geocodeservice.GeocodeCompletedEventArgs>(geocodeService\_GeocodeCompleted);

// Set the credentials and the geocode query, which could be an address or location.

findroute.geocodeservice.GeocodeRequest geocodeRequest = new findroute.geocodeservice.GeocodeRequest();

geocodeRequest.Credentials = new Credentials();

geocodeRequest.Credentials.ApplicationId = ((ApplicationIdCredentialsProvider)map1.CredentialsProvider).ApplicationId;

geocodeRequest.Query = strAddress;

// Make the asynchronous Geocode request, using the ‘waypoint index’ as

// the user state to track this request and allow it to be identified when the response is returned.

geocodeService.GeocodeAsync(geocodeRequest, waypointIndex);

}

// This is the global internal variable where results are stored. These are accessed later to calculate the route.

internal geocodeservice.GeocodeResult[] geocodeResults;

// This is the Geocode request callback method.

private void geocodeService\_GeocodeCompleted(object sender, geocodeservice.GeocodeCompletedEventArgs e)

{

// Retrieve the user state of this response (the ‘waypoint index’) to identify which geocode request

// it corresponds to.

int waypointIndex = System.Convert.ToInt32(e.UserState);

// Retrieve the GeocodeResult for this response and store it in the global variable geocodeResults, using

// the waypoint index to position it in the array.

geocodeResults[waypointIndex] = e.Result.Results[0];

// Look at each element in the global gecodeResults array to figure out if more geocode responses still

// need to be returned.

bool doneGeocoding = true;

foreach (geocodeservice.GeocodeResult gr in geocodeResults)

{

if (gr == null)

{

doneGeocoding = false;

}

}

// If the geocodeResults array is totally filled, then calculate the route.

if (doneGeocoding)

CalculateRoute(geocodeResults);

}

private void calculate\_Click(object sender, RoutedEventArgs e)

{

//Initialize the length of the results array. In this sample we have two waypoints.

geocodeResults = new geocodeservice.GeocodeResult[2];

// Make the two Geocode requests using the values of the text boxes. Also pass the waypoint indexes

// of these two values within the route.

Geocode(fromtxtbx.Text, 0);

Geocode(totxtbx.Text, 1);

}

private void CalculateRoute(geocodeservice.GeocodeResult[] results)

{

// Create the service variable and set the callback method using the CalculateRouteCompleted property.

routeservice.RouteServiceClient routeService = new routeservice.RouteServiceClient("BasicHttpBinding\_IRouteService");

routeService.CalculateRouteCompleted += new EventHandler<routeservice.CalculateRouteCompletedEventArgs>(routeService\_CalculateRouteCompleted);

// Set the token.

routeservice.RouteRequest routeRequest = new routeservice.RouteRequest();

routeRequest.Credentials = new Credentials();

routeRequest.Credentials.ApplicationId = "ApgLkoHIG4rNShRJAxMMNettsv6SWs3eP8OchozFS89Vex7BRHsSbCr31HkvYK-d";

// Return the route points so the route can be drawn.

routeRequest.Options = new routeservice.RouteOptions();

routeRequest.Options.RoutePathType = routeservice.RoutePathType.Points;

// Set the waypoints of the route to be calculated using the Geocode Service results stored in the geocodeResults variable.

routeRequest.Waypoints = new System.Collections.ObjectModel.ObservableCollection<routeservice.Waypoint>();

foreach (geocodeservice.GeocodeResult result in results)

{

routeRequest.Waypoints.Add(GeocodeResultToWaypoint(result));

}

// Make the CalculateRoute asnychronous request.

routeService.CalculateRouteAsync(routeRequest);

}

private routeservice.Waypoint GeocodeResultToWaypoint(geocodeservice.GeocodeResult result)

{

routeservice.Waypoint waypoint = new routeservice.Waypoint();

waypoint.Description = result.DisplayName;

waypoint.Location = new Location();

waypoint.Location.Latitude = result.Locations[0].Latitude;

waypoint.Location.Longitude = result.Locations[0].Longitude;

return waypoint;

}

private void routeService\_CalculateRouteCompleted(object sender, routeservice.CalculateRouteCompletedEventArgs e)

{

// If the route calculate was a success and contains a route, then draw the route on the map.

if ((e.Result.ResponseSummary.StatusCode == routeservice.ResponseStatusCode.Success) & (e.Result.Result.Legs.Count != 0))

{

// Set properties of the route line you want to draw.

Color routeColor = Colors.Blue;

SolidColorBrush routeBrush = new SolidColorBrush(routeColor);

MapPolyline routeLine = new MapPolyline();

routeLine.Locations = new LocationCollection();

routeLine.Stroke = routeBrush;

routeLine.Opacity = 0.65;

routeLine.StrokeThickness = 5.0;

// Retrieve the route points that define the shape of the route.

foreach (Location p in e.Result.Result.RoutePath.Points)

{

routeLine.Locations.Add(new Location { Latitude = p.Latitude, Longitude = p.Longitude });

}

// Add a map layer in which to draw the route.

MapLayer myRouteLayer = new MapLayer();

map1.Children.Add(myRouteLayer);

// Add the route line to the new layer.

myRouteLayer.Children.Add(routeLine);

// Figure the rectangle which encompasses the route. This is used later to set the map view.

double centerlatitude = (routeLine.Locations[0].Latitude + routeLine.Locations[routeLine.Locations.Count - 1].Latitude) / 2;

double centerlongitude = (routeLine.Locations[0].Longitude + routeLine.Locations[routeLine.Locations.Count - 1].Longitude) / 2;

Location centerloc = new Location();

centerloc.Latitude = centerlatitude;

centerloc.Longitude = centerlongitude;

double north, south, east, west;

if ((routeLine.Locations[0].Latitude > 0) && (routeLine.Locations[routeLine.Locations.Count - 1].Latitude > 0))

{

north = routeLine.Locations[0].Latitude > routeLine.Locations[routeLine.Locations.Count - 1].Latitude ? routeLine.Locations[0].Latitude : routeLine.Locations[routeLine.Locations.Count - 1].Latitude;

south = routeLine.Locations[0].Latitude < routeLine.Locations[routeLine.Locations.Count - 1].Latitude ? routeLine.Locations[0].Latitude : routeLine.Locations[routeLine.Locations.Count - 1].Latitude;

}

else

{

north = routeLine.Locations[0].Latitude < routeLine.Locations[routeLine.Locations.Count - 1].Latitude ? routeLine.Locations[0].Latitude : routeLine.Locations[routeLine.Locations.Count - 1].Latitude;

south = routeLine.Locations[0].Latitude > routeLine.Locations[routeLine.Locations.Count - 1].Latitude ? routeLine.Locations[0].Latitude : routeLine.Locations[routeLine.Locations.Count - 1].Latitude;

}

if ((routeLine.Locations[0].Longitude < 0) && (routeLine.Locations[routeLine.Locations.Count - 1].Longitude < 0))

{

west = routeLine.Locations[0].Longitude < routeLine.Locations[routeLine.Locations.Count - 1].Longitude ? routeLine.Locations[0].Longitude : routeLine.Locations[routeLine.Locations.Count - 1].Longitude;

east = routeLine.Locations[0].Longitude > routeLine.Locations[routeLine.Locations.Count - 1].Longitude ? routeLine.Locations[0].Longitude : routeLine.Locations[routeLine.Locations.Count - 1].Longitude;

}

else

{

west = routeLine.Locations[0].Longitude > routeLine.Locations[routeLine.Locations.Count - 1].Longitude ? routeLine.Locations[0].Longitude : routeLine.Locations[routeLine.Locations.Count - 1].Longitude;

east = routeLine.Locations[0].Longitude < routeLine.Locations[routeLine.Locations.Count - 1].Longitude ? routeLine.Locations[0].Longitude : routeLine.Locations[routeLine.Locations.Count - 1].Longitude;

}

// For each geocode result (which are the waypoints of the route), draw a dot on the map.

foreach (geocodeservice.GeocodeResult gr in geocodeResults)

{

Ellipse point = new Ellipse();

point.Width = 10;

point.Height = 10;

point.Fill = new SolidColorBrush(Colors.Red);

point.Opacity = 0.65;

location.Latitude = gr.Locations[0].Latitude;

location.Longitude = gr.Locations[0].Longitude;

MapLayer.SetPosition(point, location);

MapLayer.SetPositionOrigin(point, PositionOrigin.Center);

// Add the drawn point to the route layer.

myRouteLayer.Children.Add(point);

}

// Set the map view using the rectangle which bounds the rendered route.

//map1.SetView(rect);

double latitude = 0.0;

double longtitude = 0.0;

map1.SetView(location, 12);

map1.Center = location;

GeoCoordinate CurrentLocCoordinate = new System.Device.Location.GeoCoordinate(latitude, longtitude);

}

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

double zoom;

zoom = map1.ZoomLevel;

map1.ZoomLevel = ++zoom;

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

double zoom;

zoom = map1.ZoomLevel;

map1.ZoomLevel = --zoom;

}

private void view\_Click(object sender, RoutedEventArgs e)

{

if (map1.Mode is RoadMode)

{

map1.Mode = new AerialMode(true);

}

else

{

map1.Mode = new RoadMode();

}

}

}

}

<my:Map Height="482" HorizontalAlignment="Left" Margin="6,6,0,0" Name="map1" VerticalAlignment="Top" Width="444" CredentialsProvider= "ApgLkoHIG4rNShRJAxMMNettsv6SWs3eP8OchozFS89Vex7BRHsSbCr31HkvYK-d"/>