# Map Visualization with Azure

2 simple components, all routes and route details.

## All Routes:

The All Routes UI is shown below:

A screenshot of a computer

Description automatically generated

The code behind is below. Note it pulls from a mySqlConnection object and if you click the route link, the URL is “/getRoute?startTime=”, which I will discuss next.

private async Task defaultPage(HttpContext context)

{

await HTMLHead(context, false, null);

await context.Response.WriteAsync("<TR>" +

"<TD></TD>" +

"<TD align=\"center\"><B>Vehicle</B></TD>" +

"<TD align=\"center\"><B>Latitude</B></TD>" +

"<TD align=\"center\"><B>Longitude</B></TD>" +

"<TD align=\"center\"><B>Start Time</B></TD>" +

"<TD align=\"center\"><B>Points</B></TD>" +

"<TD align=\"center\"><B></B></TD>" +

"</TR>");

/\*

\* mysql> describe routes;

+--------------------+----------------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------------+----------------------+------+-----+---------+----------------+

| id | smallint(5) unsigned | NO | PRI | NULL | auto\_increment |

| deviceId | varchar(10) | NO | | NULL | |

| latitude | float(9,6) | NO | | NULL | |

| longitude | float(9,6) | NO | | NULL | |

| satelliteCount | tinyint(4) | YES | | NULL | |

| altitudeM | smallint(6) | YES | | NULL | |

| currentDateTimeUTC | datetime | YES | | NULL | |

| startDateTimeUTC | datetime | YES | | NULL | |

| groundSpeedMPH | tinyint(4) | YES | | NULL | |

| stop | smallint(6) | YES | | NULL | |

+--------------------+----------------------+------+-----+---------+----------------+

10 rows in set (0.00 sec)

\*/

await mySqlConnection.OpenAsync();

string queryAll = "select distinct vroutes.deviceId, vroutes.latitude, vroutes.longitude, vroutes.currentDateTimeUTC, vroutes.startDateTimeUTC, stops.stops from vroutes " +

"JOIN(select distinct startDateTimeUTC, count(\*) as stops from vroutes GROUP BY startDateTimeUTC) stops " +

"ON vroutes.startDateTimeUTC = stops.startDateTimeUTC where vroutes.stop = 0 order by vroutes.startDateTimeUTC ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader;

myReader = myCommand.ExecuteReader();

while (myReader.Read())

{

string deviceId = myReader.GetString("deviceId");

float latitude = myReader.GetFloat("latitude");

float longitude = myReader.GetFloat("longitude");

DateTime currentDateTimeUTC = myReader.GetDateTime("startDateTimeUTC");

DateTime currentDateTime = new DateTime();

if (System.Environment.OSVersion.Platform == PlatformID.Win32NT)

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("Central Standard Time"));

} else

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("CST6CDT"));

}

int stops = myReader.GetInt16("stops");

await context.Response.WriteAsync("<TR>" +

"<TD><a href=\"/getRoute?startTime=" + currentDateTimeUTC.ToString() + "&deviceId=" + deviceId +"\">route</a></TD>" +

"<TD align=\"center\">" + deviceId + "</TD>" +

"<TD align=\"center\">" + latitude.ToString() + "</TD>" +

"<TD align=\"center\">" + longitude.ToString() + "</TD>" +

"<TD>" + currentDateTime.ToShortDateString() + " " +

string.Format("{0:hh:mm tt}", currentDateTime) + " CST</TD>" +

"<TD align=\"center\">" + stops.ToString() + "</TD>" +

"<TD><a href=\"deleteRoute?startTime=" + currentDateTimeUTC.ToString() + "&deviceId=" + deviceId + "\">delete route</a></TD>" +

"</TR>");

}

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

await HTMLEnd(context);

}

## Route Details:

The Route Details UI is shown below:

A picture containing text, map

Description automatically generated

If you scroll down on the page you see the specific stops, lat long and speed:

A close up of a map

Description automatically generated

Using AJAX, there are three commands getRoute + getMapJS (UI above) and getRouteJSON, which is used to render the map. For getMapJS, you will need an Azure Maps subscription key.

From a web browser perspective, here is what is returned. Note the simplicity of the GetMap() AJAX function. This JavaSCRIPT is all generated by getMapJS()

<HTML><HEAD>

<style>html, body, table { width: 100%; height: 100%; padding: 0; margin: 0;} #myMap {width: 100%; height: 100%; }} </style>

<!-- Add references to JQuery. -->

<script src="https://code.jquery.com/jquery-3.1.1.js" integrity="sha256-16cdPddA6VdVInumRGo6IbivbERE8p7CQR3HzTBuELA=" crossorigin="anonymous"></script>

<!-- Add references to the Azure Maps Map control JavaScript and CSS files. -->

<link rel="stylesheet" href="https://atlas.microsoft.com/sdk/javascript/mapcontrol/2/atlas.min.css" type="text/css">

<script src="https://atlas.microsoft.com/sdk/javascript/mapcontrol/2/atlas.min.js"></script>

<!-- Add a reference to the Azure Maps Services Module JavaScript file. -->

<script src="https://atlas.microsoft.com/sdk/javascript/service/2/atlas-service.min.js"></script>

<script type="text/javascript">

function GetMap(){

var map = new atlas.Map('myMap', {

view: 'Auto',

authOptions: {

authType: 'subscriptionKey',

subscriptionKey: 'YqprREMOVEDrP1Qmew'

}

});

map.setStyle({renderWorldCopies: true, showBuildingModels: true, showFeedbackLink: false, showLogo: false});

map.events.add('ready', function () {

var dataSource = new atlas.source.DataSource();

var dataSourcePins = new atlas.source.DataSource();

map.sources.add(dataSource);

map.sources.add(dataSourcePins);

$.getJSON('/getRouteJSON?startTime=08/03/2020 02:17:11&deviceId=lincoln', function(data) {

var latMin = data[0].latitude;

var latMax = data[0].latitude;

var lonMin = data[0].longititude;

var lonMax = data[0].longititude;

var marker = [];

var counter = 0;

for (var i = 1; i < data.length; i++) {

if (data[i].longititude > lonMax) { lonMax = data[i].longititude; }

if (data[i].longititude < lonMin) { lonMin = data[i].longititude; }

if (data[i].latitude > latMax) { latMax = data[i].latitude; }

if (data[i].latitude < latMin) { latMin = data[i].latitude; }

if (counter % 4 ) { dataSourcePins.add(new atlas.data.Feature(new atlas.data.Point([data[i].longititude, data[i].latitude]), { text: data[i].groundSpeedMPH })); }

dataSource.add(new atlas.data.LineString([[data[i-1].longititude,data[i-1].latitude], [data[i].longititude,data[i].latitude]]));

counter = counter + 1;

}

console.log(lonMin, latMin, lonMax, latMax);

map.setCamera({ bounds: [lonMin, latMin, lonMax, latMax], padding: 10});

map.layers.add(new atlas.layer.SymbolLayer(dataSourcePins, null));

map.layers.add(new atlas.layer.LineLayer(dataSource, null, {

strokeColor: 'blue',

strokeWidth: 5

}));

map.markers.add(marker);

map.resize();

});

});

}

</script>

</HEAD><BODY onload="GetMap()"><TABLE align="center"><div id="myMap"></div></TABLE></BODY></HTML>

**CSharp Functions:**

private async Task getRoute(HttpContext context)

{

DateTime startTime = DateTime.Parse(context.Request.Query["startTime"].ToString());

string deviceIdLocal = context.Request.Query["deviceId"].ToString();

await HTMLHead(context, true, getMapJS(context.Request.Query["startTime"].ToString(), context.Request.Query["deviceId"].ToString()));

//await context.Response.WriteAsync("<TR><TD colspan=6><div id=\"myMap\"></div></TD></TR>");

await context.Response.WriteAsync("<div id=\"myMap\"></div>");

await context.Response.WriteAsync("<TR>" +

"<TD align=\"center\"><B>Vehicle</B></TD>" +

"<TD align=\"center\"><B>Latitude</B></TD>" +

"<TD align=\"center\"><B>Longitude</B></TD>" +

"<TD align=\"center\"><B>stop</B></TD>" +

"<TD align=\"center\"><B>speed</B></TD>" +

"<TD align=\"center\"><B>Time</B></TD>" +

"</TR>");

await mySqlConnection.OpenAsync();

string queryAll = "select distinct deviceId, latitude, " +

"longitude, currentDateTimeUTC," +

"groundSpeedMPH, stop " +

"from vroutes where startDateTimeUTC = '" +

string.Format("{0:yyyy-MM-dd HH:mm:ss}", startTime) + "' " +

"and deviceId = '" + deviceIdLocal + "' " +

"order by currentDateTimeUTC ASC, stop ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader;

myReader = myCommand.ExecuteReader();

bool currentRoute = true;

int laststop = 0;

while (myReader.Read() && currentRoute == true)

{

string deviceId = myReader.GetString("deviceId");

float latitude = myReader.GetFloat("latitude");

float longitude = myReader.GetFloat("longitude");

DateTime currentDateTimeUTC = myReader.GetDateTime("currentDateTimeUTC");

DateTime currentDateTime = new DateTime();

if (System.Environment.OSVersion.Platform == PlatformID.Win32NT)

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("Central Standard Time"));

}

else

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("CST6CDT"));

}

int groundSpeedMPH = myReader.GetInt16("groundSpeedMPH");

int stop = myReader.GetInt16("stop");

if (stop < laststop)

{

break;

}

string URL = "https://www.bing.com/maps?q=" + latitude.ToString() + "%2C" + longitude.ToString();

await context.Response.WriteAsync("<TR>" +

"<TD>" + deviceId + "</TD>" +

"<TD><a href=\"" + URL + "\">" + latitude.ToString() + "</a></TD>" +

"<TD><a href=\"" + URL + "\">" + longitude.ToString() + "</a></TD>" +

"<TD align=\"center\">" + stop.ToString() + "</TD>" +

"<TD align=\"center\">" + groundSpeedMPH.ToString() + "</TD>" +

"<TD>" + currentDateTime.ToShortDateString() + " " +

string.Format("{0:hh:mm tt}", currentDateTime) + " CST</TD></TR>");

laststop = stop;

}

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

await HTMLEnd(context);

}

private string getMapJS(string currentDateTimeUTC, string deviceId)

{

// reference: https://docs.microsoft.com/en-us/azure/azure-maps/map-add-line-layer

StringBuilder csText = new StringBuilder();

csText.AppendLine("<script type=\"text/javascript\">");

csText.AppendLine("function GetMap(){");

csText.AppendLine("var map = new atlas.Map('myMap', {");

csText.AppendLine(" view: 'Auto',");

csText.AppendLine(" authOptions: {");

csText.AppendLine(" authType: 'subscriptionKey',");

csText.AppendLine(" subscriptionKey: '" + subscriptionKey + "'");

csText.AppendLine(" }"); // close the authOptions

csText.AppendLine("});"); // close the var map

csText.AppendLine("map.setStyle({renderWorldCopies: true, showBuildingModels: true, showFeedbackLink: false, showLogo: false});");

csText.AppendLine("map.events.add('ready', function () {");

csText.AppendLine(" var dataSource = new atlas.source.DataSource();");

csText.AppendLine(" var dataSourcePins = new atlas.source.DataSource();");

csText.AppendLine(" map.sources.add(dataSource);");

csText.AppendLine(" map.sources.add(dataSourcePins);");

csText.AppendLine(" $.getJSON('/getRouteJSON?startTime=" + currentDateTimeUTC + "&deviceId=" + deviceId +"', function(data) {");

csText.AppendLine(" var latMin = data[0].latitude;");

csText.AppendLine(" var latMax = data[0].latitude;");

csText.AppendLine(" var lonMin = data[0].longititude;");

csText.AppendLine(" var lonMax = data[0].longititude;");

csText.AppendLine(" var marker = [];");

csText.AppendLine(" var counter = 0;");

csText.AppendLine(" for (var i = 1; i < data.length; i++) {");

csText.AppendLine(" if (data[i].longititude > lonMax) { lonMax = data[i].longititude; }");

csText.AppendLine(" if (data[i].longititude < lonMin) { lonMin = data[i].longititude; }");

csText.AppendLine(" if (data[i].latitude > latMax) { latMax = data[i].latitude; }");

csText.AppendLine(" if (data[i].latitude < latMin) { latMin = data[i].latitude; }");

csText.AppendLine(" if (counter % 4 ) { dataSourcePins.add(new atlas.data.Feature(new atlas.data.Point([data[i].longititude, data[i].latitude]), { text: data[i].groundSpeedMPH })); }");

csText.AppendLine(" dataSource.add(new atlas.data.LineString([[data[i-1].longititude,data[i-1].latitude], [data[i].longititude,data[i].latitude]]));");

csText.AppendLine(" counter = counter + 1;");

csText.AppendLine(" }"); // close the for loop

csText.AppendLine(" console.log(lonMin, latMin, lonMax, latMax);"); // close the for loop

csText.AppendLine(" map.setCamera({ bounds: [lonMin, latMin, lonMax, latMax], padding: 10});");

csText.AppendLine(" map.layers.add(new atlas.layer.SymbolLayer(dataSourcePins, null));");

csText.AppendLine(" map.layers.add(new atlas.layer.LineLayer(dataSource, null, {");

csText.AppendLine(" strokeColor: 'blue',");

csText.AppendLine(" strokeWidth: 5");

csText.AppendLine(" }));"); // close the map.layers.add

csText.AppendLine(" map.markers.add(marker);");

csText.AppendLine(" map.resize();");

csText.AppendLine(" });"); // close the $.getJSON(

csText.AppendLine(" });"); // close the map.events.add

csText.AppendLine("}"); // close the getMap()

csText.AppendLine("</script>");

return csText.ToString();

}

private async Task getRouteJSON(HttpContext context)

{

routeArray.Clear();

DateTime startTime = DateTime.Parse(context.Request.Query["startTime"].ToString());

string deviceIdLocal = context.Request.Query["deviceId"].ToString();

await mySqlConnection.OpenAsync();

string queryAll = "select distinct deviceId, latitude, " +

"longitude, currentDateTimeUTC," +

"groundSpeedMPH, stop " +

"from vroutes where startDateTimeUTC = '" +

string.Format("{0:yyyy-MM-dd HH:mm:ss}", startTime) + "' " +

"and deviceId = '" + deviceIdLocal + "' " +

"order by currentDateTimeUTC ASC, stop ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader = myCommand.ExecuteReader();

while (myReader.Read())

{

JObject entry = new JObject();

entry["deviceId"] = myReader.GetString("deviceId");

entry["latitude"] = myReader.GetFloat("latitude");

entry["longititude"] = myReader.GetFloat("longitude");

entry["groundSpeedMPH"] = myReader.GetInt16("groundSpeedMPH");

entry["stop"] = myReader.GetInt16("stop");

routeArray.Add(entry);

}

context.Response.Clear();

context.Response.ContentType = "application/json; charset=utf-8";

await context.Response.WriteAsync(routeArray.ToString());

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

}

## All of the CSharp Code:

using System;

using System.Text;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using MySql.Data.MySqlClient;

using Newtonsoft.Json.Linq;

namespace WebApplication

{

public class Startup

{

private const string mysqlconnectionstring = "server=192.168.15.200;uid=REMOVED;pwd=REMOIVED;database=autotracker";

/\* CREATE TABLE routes (id smallint unsigned not null auto\_increment, deviceId varchar(10) not null, latitude FLOAT(9,6) not null, longitude FLOAT(9,6) not null, satelliteCount TINYINT, altitudeM SMALLINT, currentDateTimeUTC DATETIME, groundSpeedMPH TINYINT, stop SMALLINT, PRIMARY KEY ( id ));

\*/

static MySqlConnection mySqlConnection;

static JArray routeArray = new JArray();

static string subscriptionKey = "YqREMOVEDmew";

// This method gets called by the runtime. Use this method to add services to the container.

// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940

public void ConfigureServices(IServiceCollection services)

{

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

mySqlConnection = new MySqlConnection(mysqlconnectionstring);

app.UseRouting();

app.UseEndpoints(endpoints =>

{

endpoints.MapGet("/", defaultPage);

endpoints.MapGet("/getRoute", getRoute);

endpoints.MapGet("/getRouteJSON", getRouteJSON);

endpoints.MapGet("/deleteRoute", deleteRoute);

});

}

private async Task getRouteJSON(HttpContext context)

{

routeArray.Clear();

DateTime startTime = DateTime.Parse(context.Request.Query["startTime"].ToString());

string deviceIdLocal = context.Request.Query["deviceId"].ToString();

await mySqlConnection.OpenAsync();

string queryAll = "select distinct deviceId, latitude, " +

"longitude, currentDateTimeUTC," +

"groundSpeedMPH, stop " +

"from vroutes where startDateTimeUTC = '" +

string.Format("{0:yyyy-MM-dd HH:mm:ss}", startTime) + "' " +

"and deviceId = '" + deviceIdLocal + "' " +

"order by currentDateTimeUTC ASC, stop ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader = myCommand.ExecuteReader();

while (myReader.Read())

{

JObject entry = new JObject();

entry["deviceId"] = myReader.GetString("deviceId");

entry["latitude"] = myReader.GetFloat("latitude");

entry["longititude"] = myReader.GetFloat("longitude");

entry["groundSpeedMPH"] = myReader.GetInt16("groundSpeedMPH");

entry["stop"] = myReader.GetInt16("stop");

routeArray.Add(entry);

}

context.Response.Clear();

context.Response.ContentType = "application/json; charset=utf-8";

await context.Response.WriteAsync(routeArray.ToString());

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

}

private async Task deleteRoute(HttpContext context)

{

DateTime startTime = DateTime.Parse(context.Request.Query["startTime"].ToString());

string deviceIdLocal = context.Request.Query["deviceId"].ToString();

await mySqlConnection.OpenAsync();

string queryAll = "delete from routes where startDateTimeUTC = '" +

string.Format("{0:yyyy-MM-dd HH:mm:ss}", startTime) + "' " +

"and deviceId = '" + deviceIdLocal + "';";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

await myCommand.ExecuteNonQueryAsync();

await mySqlConnection.CloseAsync();

await defaultPage(context);

}

private async Task getRoute(HttpContext context)

{

DateTime startTime = DateTime.Parse(context.Request.Query["startTime"].ToString());

string deviceIdLocal = context.Request.Query["deviceId"].ToString();

await HTMLHead(context, true, getMapJS(context.Request.Query["startTime"].ToString(), context.Request.Query["deviceId"].ToString()));

//await context.Response.WriteAsync("<TR><TD colspan=6><div id=\"myMap\"></div></TD></TR>");

await context.Response.WriteAsync("<div id=\"myMap\"></div>");

await context.Response.WriteAsync("<TR>" +

"<TD align=\"center\"><B>Vehicle</B></TD>" +

"<TD align=\"center\"><B>Latitude</B></TD>" +

"<TD align=\"center\"><B>Longitude</B></TD>" +

"<TD align=\"center\"><B>stop</B></TD>" +

"<TD align=\"center\"><B>speed</B></TD>" +

"<TD align=\"center\"><B>Time</B></TD>" +

"</TR>");

await mySqlConnection.OpenAsync();

string queryAll = "select distinct deviceId, latitude, " +

"longitude, currentDateTimeUTC," +

"groundSpeedMPH, stop " +

"from vroutes where startDateTimeUTC = '" +

string.Format("{0:yyyy-MM-dd HH:mm:ss}", startTime) + "' " +

"and deviceId = '" + deviceIdLocal + "' " +

"order by currentDateTimeUTC ASC, stop ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader;

myReader = myCommand.ExecuteReader();

bool currentRoute = true;

int laststop = 0;

while (myReader.Read() && currentRoute == true)

{

string deviceId = myReader.GetString("deviceId");

float latitude = myReader.GetFloat("latitude");

float longitude = myReader.GetFloat("longitude");

DateTime currentDateTimeUTC = myReader.GetDateTime("currentDateTimeUTC");

DateTime currentDateTime = new DateTime();

if (System.Environment.OSVersion.Platform == PlatformID.Win32NT)

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("Central Standard Time"));

}

else

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("CST6CDT"));

}

int groundSpeedMPH = myReader.GetInt16("groundSpeedMPH");

int stop = myReader.GetInt16("stop");

if (stop < laststop)

{

break;

}

string URL = "https://www.bing.com/maps?q=" + latitude.ToString() + "%2C" + longitude.ToString();

await context.Response.WriteAsync("<TR>" +

"<TD>" + deviceId + "</TD>" +

"<TD><a href=\"" + URL + "\">" + latitude.ToString() + "</a></TD>" +

"<TD><a href=\"" + URL + "\">" + longitude.ToString() + "</a></TD>" +

"<TD align=\"center\">" + stop.ToString() + "</TD>" +

"<TD align=\"center\">" + groundSpeedMPH.ToString() + "</TD>" +

"<TD>" + currentDateTime.ToShortDateString() + " " +

string.Format("{0:hh:mm tt}", currentDateTime) + " CST</TD></TR>");

laststop = stop;

}

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

await HTMLEnd(context);

}

private string getMapJS(string currentDateTimeUTC, string deviceId)

{

// reference: https://docs.microsoft.com/en-us/azure/azure-maps/map-add-line-layer

StringBuilder csText = new StringBuilder();

csText.AppendLine("<script type=\"text/javascript\">");

csText.AppendLine("function GetMap(){");

csText.AppendLine("var map = new atlas.Map('myMap', {");

csText.AppendLine(" view: 'Auto',");

//csText.AppendLine(" center: [-73.985708, 40.75773], zoom: 11,");

csText.AppendLine(" authOptions: {");

csText.AppendLine(" authType: 'subscriptionKey',");

csText.AppendLine(" subscriptionKey: '" + subscriptionKey + "'");

csText.AppendLine(" }"); // close the authOptions

csText.AppendLine("});"); // close the var map

csText.AppendLine("map.setStyle({renderWorldCopies: true, showBuildingModels: true, showFeedbackLink: false, showLogo: false});");

csText.AppendLine("map.events.add('ready', function () {");

csText.AppendLine(" var dataSource = new atlas.source.DataSource();");

csText.AppendLine(" var dataSourcePins = new atlas.source.DataSource();");

csText.AppendLine(" map.sources.add(dataSource);");

csText.AppendLine(" map.sources.add(dataSourcePins);");

csText.AppendLine(" $.getJSON('/getRouteJSON?startTime=" + currentDateTimeUTC + "&deviceId=" + deviceId +"', function(data) {");

csText.AppendLine(" var latMin = data[0].latitude;");

csText.AppendLine(" var latMax = data[0].latitude;");

csText.AppendLine(" var lonMin = data[0].longititude;");

csText.AppendLine(" var lonMax = data[0].longititude;");

csText.AppendLine(" var marker = [];");

csText.AppendLine(" var counter = 0;");

csText.AppendLine(" for (var i = 1; i < data.length; i++) {");

csText.AppendLine(" if (data[i].longititude > lonMax) { lonMax = data[i].longititude; }");

csText.AppendLine(" if (data[i].longititude < lonMin) { lonMin = data[i].longititude; }");

csText.AppendLine(" if (data[i].latitude > latMax) { latMax = data[i].latitude; }");

csText.AppendLine(" if (data[i].latitude < latMin) { latMin = data[i].latitude; }");

//csText.AppendLine(" var marker = new atlas.HtmlMarker({color: 'DodgerBlue', text: data[i].groundSpeedMPH, position: [data[i].longititude, data[i].latitude]});");

//csText.AppendLine(" if (counter % 4 ) { ");

//csText.AppendLine(" marker.push(new atlas.HtmlMarker({color: 'DodgerBlue', text: '1'}));");

//csText.AppendLine(" }");

//csText.AppendLine(" dataSourcePins.add([new atlas.Shape(new atlas.data.Point([data[i].longititude, data[i].latitude]))]);");

csText.AppendLine(" if (counter % 4 ) { dataSourcePins.add(new atlas.data.Feature(new atlas.data.Point([data[i].longititude, data[i].latitude]), { text: data[i].groundSpeedMPH })); }");

csText.AppendLine(" dataSource.add(new atlas.data.LineString([[data[i-1].longititude,data[i-1].latitude], [data[i].longititude,data[i].latitude]]));");

csText.AppendLine(" counter = counter + 1;");

csText.AppendLine(" }"); // close the for loop

csText.AppendLine(" console.log(lonMin, latMin, lonMax, latMax);"); // close the for loop

csText.AppendLine(" map.setCamera({ bounds: [lonMin, latMin, lonMax, latMax], padding: 10});");

csText.AppendLine(" map.layers.add(new atlas.layer.SymbolLayer(dataSourcePins, null));");

csText.AppendLine(" map.layers.add(new atlas.layer.LineLayer(dataSource, null, {");

csText.AppendLine(" strokeColor: 'blue',");

csText.AppendLine(" strokeWidth: 5");

csText.AppendLine(" }));"); // close the map.layers.add

csText.AppendLine(" map.markers.add(marker);");

csText.AppendLine(" map.resize();");

csText.AppendLine(" });"); // close the $.getJSON(

csText.AppendLine(" });"); // close the map.events.add

csText.AppendLine("}"); // close the getMap()

csText.AppendLine("</script>");

return csText.ToString();

}

private async Task defaultPage(HttpContext context)

{

await HTMLHead(context, false, null);

await context.Response.WriteAsync("<TR>" +

"<TD></TD>" +

"<TD align=\"center\"><B>Vehicle</B></TD>" +

"<TD align=\"center\"><B>Latitude</B></TD>" +

"<TD align=\"center\"><B>Longitude</B></TD>" +

"<TD align=\"center\"><B>Start Time</B></TD>" +

"<TD align=\"center\"><B>Points</B></TD>" +

"<TD align=\"center\"><B></B></TD>" +

"</TR>");

/\*

\* mysql> describe routes;

+--------------------+----------------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------------+----------------------+------+-----+---------+----------------+

| id | smallint(5) unsigned | NO | PRI | NULL | auto\_increment |

| deviceId | varchar(10) | NO | | NULL | |

| latitude | float(9,6) | NO | | NULL | |

| longitude | float(9,6) | NO | | NULL | |

| satelliteCount | tinyint(4) | YES | | NULL | |

| altitudeM | smallint(6) | YES | | NULL | |

| currentDateTimeUTC | datetime | YES | | NULL | |

| startDateTimeUTC | datetime | YES | | NULL | |

| groundSpeedMPH | tinyint(4) | YES | | NULL | |

| stop | smallint(6) | YES | | NULL | |

+--------------------+----------------------+------+-----+---------+----------------+

10 rows in set (0.00 sec)

\*/

await mySqlConnection.OpenAsync();

string queryAll = "select distinct vroutes.deviceId, vroutes.latitude, vroutes.longitude, vroutes.currentDateTimeUTC, vroutes.startDateTimeUTC, stops.stops from vroutes " +

"JOIN(select distinct startDateTimeUTC, count(\*) as stops from vroutes GROUP BY startDateTimeUTC) stops " +

"ON vroutes.startDateTimeUTC = stops.startDateTimeUTC where vroutes.stop = 0 order by vroutes.startDateTimeUTC ASC;";

MySqlCommand myCommand = new MySqlCommand(queryAll, mySqlConnection);

MySqlDataReader myReader;

myReader = myCommand.ExecuteReader();

while (myReader.Read())

{

string deviceId = myReader.GetString("deviceId");

float latitude = myReader.GetFloat("latitude");

float longitude = myReader.GetFloat("longitude");

DateTime currentDateTimeUTC = myReader.GetDateTime("startDateTimeUTC");

DateTime currentDateTime = new DateTime();

if (System.Environment.OSVersion.Platform == PlatformID.Win32NT)

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("Central Standard Time"));

} else

{

currentDateTime = TimeZoneInfo.ConvertTimeFromUtc(myReader.GetDateTime("currentDateTimeUTC"), TimeZoneInfo.FindSystemTimeZoneById("CST6CDT"));

}

int stops = myReader.GetInt16("stops");

await context.Response.WriteAsync("<TR>" +

"<TD><a href=\"/getRoute?startTime=" + currentDateTimeUTC.ToString() + "&deviceId=" + deviceId +"\">route</a></TD>" +

"<TD align=\"center\">" + deviceId + "</TD>" +

"<TD align=\"center\">" + latitude.ToString() + "</TD>" +

"<TD align=\"center\">" + longitude.ToString() + "</TD>" +

"<TD>" + currentDateTime.ToShortDateString() + " " +

string.Format("{0:hh:mm tt}", currentDateTime) + " CST</TD>" +

"<TD align=\"center\">" + stops.ToString() + "</TD>" +

"<TD><a href=\"deleteRoute?startTime=" + currentDateTimeUTC.ToString() + "&deviceId=" + deviceId + "\">delete route</a></TD>" +

"</TR>");

}

// always call Close when done reading.

await myReader.CloseAsync();

await myReader.DisposeAsync();

await mySqlConnection.CloseAsync();

await HTMLEnd(context);

}

private async Task HTMLHead(HttpContext context, bool addMap, string script)

{

await context.Response.WriteAsync("<HTML><HEAD>\r");

if (addMap) {

//await context.Response.WriteAsync("<link rel=\"stylesheet\" type=\"text/css\" href=\"./StyleSheet.css\">\r");

await context.Response.WriteAsync("<style>html, body, table { width: 100%; height: 100%; padding: 0; margin: 0;} #myMap {width: 100%; height: 100%; }} </style>\r");

await context.Response.WriteAsync("<!-- Add references to JQuery. -->\r");

await context.Response.WriteAsync("<script src=\"https://code.jquery.com/jquery-3.1.1.js\" integrity=\"sha256-16cdPddA6VdVInumRGo6IbivbERE8p7CQR3HzTBuELA=\" crossorigin=\"anonymous\"></script>\r");

// adding Azure Maps files

await context.Response.WriteAsync("\r");

await context.Response.WriteAsync("<!-- Add references to the Azure Maps Map control JavaScript and CSS files. -->\r");

await context.Response.WriteAsync("<link rel=\"stylesheet\" href=\"https://atlas.microsoft.com/sdk/javascript/mapcontrol/2/atlas.min.css\" type=\"text/css\">\r");

await context.Response.WriteAsync("<script src=\"https://atlas.microsoft.com/sdk/javascript/mapcontrol/2/atlas.min.js\"></script>\r");

await context.Response.WriteAsync("\r");

await context.Response.WriteAsync("<!-- Add a reference to the Azure Maps Services Module JavaScript file. -->\r");

await context.Response.WriteAsync("<script src=\"https://atlas.microsoft.com/sdk/javascript/service/2/atlas-service.min.js\"></script>\r");

await context.Response.WriteAsync("\r");

await context.Response.WriteAsync(script + "\r");

await context.Response.WriteAsync("</HEAD><BODY onload=\"GetMap()\">");

} else {

await context.Response.WriteAsync("</HEAD><BODY>");

}

await context.Response.WriteAsync("<TABLE align=\"center\">");

}

private async Task HTMLEnd(HttpContext context)

{

await context.Response.WriteAsync("</TABLE></BODY></HTML>");

}

}

}