Customize Azure IOT Suite Dashboard

內容

[準備事項 4](#_Toc460785071)

[故障記錄圖表 5](#_Toc460785072)

[**在這一節中，我們展的是如何使用現有的圖表，修改資料，讓圖表展示我們需要的內容，因此不會改變到網頁的程式。** 5](#_Toc460785073)

[每月故障統計表 8](#_Toc460785074)

[**在這一節中，我們展的是如何新增一個新的圖表，讓圖表展示我們需要的內容；在這裡，我們使用Chart.Js作為展示層的Library，同時撰寫一個新的WebApi讓圖表呼叫以取得資料。** 8](#_Toc460785076)

[即時故障統計圖表 20](#_Toc460785077)

# 準備事項

* 需搭配IOT Suite專用的device simulator：<https://github.com/michael-chi/Azure-IOTHub/tree/master/device-simulator/iotsuite-simulator/iotsuite-simulator>
  + 此Simulator會透過telemetry傳送device metadata
* 完整程式碼在：<https://github.com/michael-chi/azure-iot-suite-customization-demo>

# 故障記錄圖表

**在這一節中，我們展的是如何使用現有的圖表，修改資料，讓圖表展示我們需要的內容，因此不會改變到網頁的程式。**

* 設備數據格式如下

public string DeviceId { get; set; }

//设备ID”

public int \_0 { get; set; }

//电压

public int \_100 { get; set; }

//电流

public int \_101 { get; set; }

//功率

public int \_102 { get; set; }

//故障代碼

public int \_0106 { get; set; }

* 用以下script建立一個故障代碼表(實際狀況您可能已經有這個故障代碼表了)

CREATE TABLE [dbo].[Table]

(

[ErrorCode] INT NOT NULL PRIMARY KEY,

[ErrorMsg] nvarchar(24) Not NULL

)

* 加入以下初始值到資料表

insert into Table(errorCode,ErrorMsg) values(1001, ‘ 电网电流过载‘)

insert into Table(errorCode,ErrorMsg) values(1002, ‘组2电压采样电路异常 ‘)

insert into Table(errorCode,ErrorMsg) values(1, ‘一般异常 ‘)

* 建立一個Data Factory或是web job將資料庫資料定期轉到blob
* 修改Stream Analytics Job michi-byddemo2-RG-Rules判斷錯誤
  + 建立一個Input型別為Reference Data名為ErrorCodeRef，指向剛剛設定的blob
  + ~~建立一個input型別為stream data，指向iot hub~~
  + ~~建立query，指令如下~~
  + 修改查詢如下

- --WITH AlarmsData AS

--(

--SELECT

-- Stream.IoTHub.ConnectionDeviceId AS DeviceId,

-- 'Temperature' as ReadingType,

-- Stream.Temperature as Reading,

-- Ref.Temperature as Threshold,

-- Ref.TemperatureRuleOutput as RuleOutput,

-- Stream.EventEnqueuedUtcTime AS [Time]

--FROM IoTTelemetryStream Stream

--JOIN DeviceRulesBlob Ref ON Stream.IoTHub.ConnectionDeviceId = Ref.DeviceID

--WHERE

-- Ref.Temperature IS NOT null AND Stream.Temperature > Ref.Temperature

--

--UNION ALL

--

--SELECT

-- Stream.IoTHub.ConnectionDeviceId AS DeviceId,

-- 'Humidity' as ReadingType,

-- Stream.Humidity as Reading,

-- Ref.Humidity as Threshold,

-- Ref.HumidityRuleOutput as RuleOutput,

-- Stream.EventEnqueuedUtcTime AS [Time]

--FROM IoTTelemetryStream Stream

--JOIN DeviceRulesBlob Ref ON Stream.IoTHub.ConnectionDeviceId = Ref.DeviceID

--WHERE

-- Ref.Humidity IS NOT null AND Stream.Humidity > Ref.Humidity

--)

WITH AlarmsData AS

(

SELECT

Stream.DeviceId AS DeviceId,

'\_0106' as ReadingType,

Stream.\_0106 as Reading,

Ref.ErrorCode as Threshold,

Ref.ErrorMsg as RuleOutput,

Stream.EventEnqueuedUtcTime AS [Time]

FROM IoTTelemetryStream Stream

Join ErrorCodeRef Ref on

Stream.\_0106 = Ref.ErrorCode

)

SELECT \*

INTO DeviceRulesMonitoring

FROM AlarmsData

SELECT \*

INTO DeviceRulesHub

FROM AlarmsData

* 執行此Stream analytics job



# 每月故障統計表



**在這一節中，我們展的是如何新增一個新的圖表，讓圖表展示我們需要的內容；在這裡，我們使用Chart.Js作為展示層的Library，同時撰寫一個新的WebApi讓圖表呼叫以取得資料。**

* 建立一個每月故障統計的資料表如下

CREATE TABLE [dbo].[TelemetrySummary] (

[DeviceId] VARCHAR (24) NOT NULL,

[ErrorCode] INT NOT NULL,

[Time] DATETIME NOT NULL,

[StatusCode] int NULL

);

* 建立一個新的Stream Analytics Job名為Byd-Telemetry-Summary
  + 建立一個Input指向IOT HUB名為DeviceData
  + 建立一個output指向SQL database名為telemetrysummary
  + 設定查詢如下

SELECT

STREAM.DeviceId AS DEVICEID,

STREAM.\_0106 as ErrorCode,

System.Timestamp time

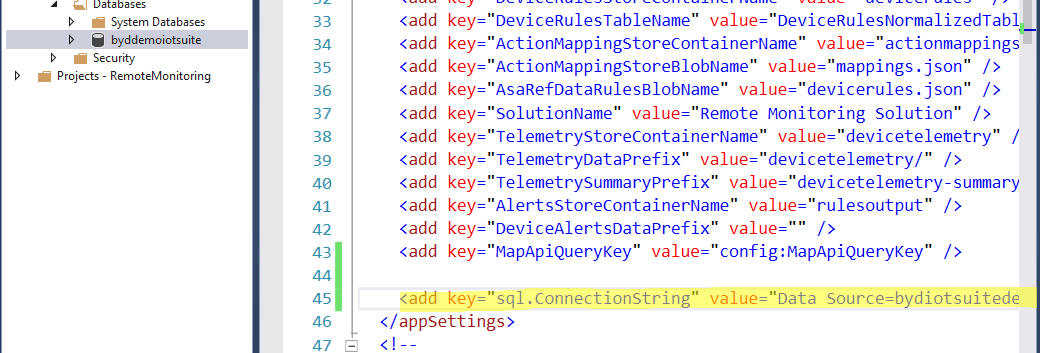
into

TelemetrySummary

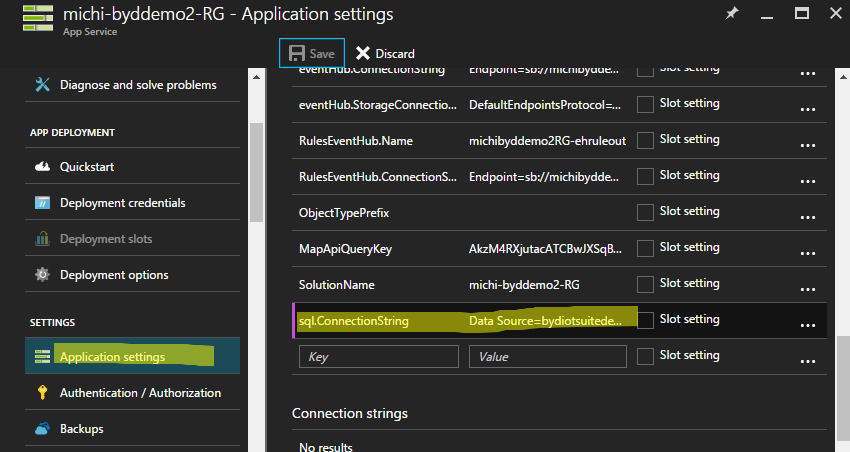
from

DeviceData STREAM

* 執行該查詢開始將故障資料轉入SQL Database
* 打開Visual Studio專案，在web.config中加上以下的appsettings



* 在Azure Portal上，打開網站，加上同樣的application setting



* 在Infrastructure專案的Models目錄新增一個BYDMonthlyErrorSummaryModel.cs

public class BYDMonthlyErrorSummaryModel

{

public string Date { get; set; }

public int ErrorCount { get; set; }

}

* 在Infrastructure專案的Models目錄新增一個BYDYearlyErrorSummaryResultModel.cs

public class BYDYearlyErrorSummaryResultModel

{

public int Year { get; set; }

public IList<BYDMonthlyErrorSummaryModel> Data { get; set; }

}

* 在Infrastructure專案的Models目錄下，新增一個BydError.cs如下

public class BYDError

{

public string DeviceId { get; set; }

public int ErrorCode { get; set; }

public DateTime Time { get; set; }

public int StatusCode { get; set; }

}

* 在Infrastructure專案的Repository目錄下，新增一個IBydErrorRepository.cs如下

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Models;

namespace Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Repository

{

public interface IBYDErrorRepository

{

Task<IEnumerable<BYDError>> GetErrorsAsync();

Task<IEnumerable<BYDError>> GetErrorsAsync(string deviceId);

Task<IEnumerable<BYDError>> GetErrorsAsync(DateTime start, DateTime end);

}

}

* 在Infrastructure專案的Repository目錄下，新增BydErrorRepository.cs如下

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.Common.Configurations;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.Common.Helpers;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.Common.Models;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Models;

using System.Data.SqlClient;

using System.Data;

using System.Text;

using Newtonsoft.Json;

namespace Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Repository

{

public class BYDErrorRepository : IBYDErrorRepository

{

private readonly IConfigurationProvider \_configurationProvider;

public BYDErrorRepository(IConfigurationProvider configurationProvider)

{

\_configurationProvider = configurationProvider;

}

public async Task<IEnumerable<BYDError>> GetErrorsAsync(DateTime start, DateTime end)

{

using (SqlConnection conn = new SqlConnection(\_configurationProvider.GetConfigurationSettingValue("sql.ConnectionString")))

{

conn.Open();

using (SqlCommand cmd = new SqlCommand("select \* from TelemetrySummary where time between @start and @end for json auto", conn))

{

cmd.Parameters.Add(new SqlParameter("@start", new SqlDateTime(start)));

cmd.Parameters.Add(new SqlParameter("@end", new SqlDateTime(end)));

using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))

{

var ds = new DataSet();

adapter.Fill(ds);

if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)

{

var sb = new StringBuilder();

foreach (DataRow row in ds.Tables[0].Rows)

{

sb.Append(row[0] as string);

}

BYDError[] errors = JsonConvert.DeserializeObject<BYDError[]>(sb.ToString());

return errors;

}

}

}

conn.Close();

}

return null;

}

public async Task<IEnumerable<BYDError>> GetErrorsAsync()

{

using (SqlConnection conn = new SqlConnection(\_configurationProvider.GetConfigurationSettingValue("sql.ConnectionString")))

{

conn.Open();

using(SqlCommand cmd = new SqlCommand("select \* from TelemetrySummary for json auto", conn))

{

using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))

{

var ds = new DataSet();

adapter.Fill(ds);

if(ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)

{

var sb = new StringBuilder();

foreach (DataRow row in ds.Tables[0].Rows)

{

sb.Append(row[0] as string);

}

BYDError[] errors = JsonConvert.DeserializeObject<BYDError[]>(sb.ToString());

return errors;

}

}

}

conn.Close();

}

return null;

}

public async Task<IEnumerable<BYDError>> GetErrorsAsync(string deviceId)

{

using (SqlConnection conn = new SqlConnection(\_configurationProvider.GetConfigurationSettingValue("sql.ConnectionString")))

{

conn.Open();

using (SqlCommand cmd = new SqlCommand("select \* from TelemetrySummary where DeviceId = @deviceId for json auto", conn))

{

cmd.Parameters.Add(new SqlParameter("@deviceId", deviceId));

using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))

{

var ds = new DataSet();

adapter.Fill(ds);

if (ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)

{

var sb = new StringBuilder();

foreach (DataRow row in ds.Tables[0].Rows)

{

sb.Append(row[0] as string);

}

BYDError[] errors = JsonConvert.DeserializeObject<BYDError[]>(sb.ToString());

return errors;

}

}

}

conn.Close();

}

return null;

}

}

}

* 在Infrastructure專案的BusinessLogic目錄，新增IBYDErrorLogic.cs 如下

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Models;

namespace Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Repository

{

public interface IBYDErrorLogic

{

Task<IEnumerable<BYDError>> GetErrorsAsync();

Task<BYDYearlyErrorSummaryResultModel> GetYearlyErrorsSummaryAsync();

Task<IEnumerable<BYDError>> GetErrorsAsync(string deviceId);

} }

* 在Infrastructure專案的BusinessLogic目錄，新增BYDErrorLogic.cs 如下

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Models;

using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.Repository;

using System.Data.SqlTypes;

namespace Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Infrastructure.BusinessLogic

{

public class BYDErrorLogic : IBYDErrorLogic

{

IBYDErrorRepository \_errorRepository = null;

public BYDErrorLogic(IBYDErrorRepository errorRepository)

{

\_errorRepository = errorRepository;

}

public async Task<IEnumerable<BYDError>> GetErrorsAsync()

{

var result = await \_errorRepository.GetErrorsAsync();

return result;

}

public async Task GetDeviceStatusAsync()

{

}

public async Task<BYDYearlyErrorSummaryResultModel> GetYearlyErrorsSummaryAsync()

{

var errors = await \_errorRepository.GetErrorsAsync(

new DateTime(DateTime.UtcNow.Year, 1, 1),

new DateTime(DateTime.UtcNow.Year, 12, 31)

);

var groups = from error in errors

group error by string.Format("{0}/{1:00}", DateTime.UtcNow.Year, error.Time.Month);

var monthlySummary = from gp in groups

select new BYDMonthlyErrorSummaryModel

{

Date = gp.Key,

ErrorCount = gp.Count()

};

var allMonthes = new int[] { 1,2,3,4,5,6,7,8,9,10,11,12};

var result = new BYDYearlyErrorSummaryResultModel

{

Year = DateTime.UtcNow.Year,

Data = monthlySummary.ToList()

};

var monthes = result.Data.Select(d => int.Parse(d.Date.Split('/')[1])).ToArray();

var missingMonthes = from month in allMonthes

where !monthes.Contains(month)

select month;

//allMonthes.SkipWhile(i => monthes.Contains(i));

var missingObj = missingMonthes.Select(m => new BYDMonthlyErrorSummaryModel

{

Date = string.Format("{0}/{1:00}",DateTime.UtcNow.Year,m),

ErrorCount = 0

});

SortedList<string, BYDMonthlyErrorSummaryModel> slist = new SortedList<string, BYDMonthlyErrorSummaryModel>();

foreach(var addedMonth in result.Data)

{

slist.Add(addedMonth.Date, addedMonth);

}

foreach(var missing in missingObj)

{

slist.Add(missing.Date, missing);

}

result.Data = slist.Select(x => x.Value).ToList();

return result;

}

public async Task<IEnumerable<BYDError>> GetErrorsAsync(string deviceId)

{

var result = await \_errorRepository.GetErrorsAsync(deviceId);

return result;

}

}

}

* 在Web專案的WebApiControllers目錄下，修改TelemetryApiController.cs
  + Constructor

public TelemetryApiController(

IDeviceTelemetryLogic deviceTelemetryLogic,

IAlertsLogic alertsLogic,

IDeviceLogic deviceLogic,

/\*Michael\*/IBYDErrorLogic bydErrorLogic,

IConfigurationProvider configProvider)

{

if (deviceTelemetryLogic == null)

{

throw new ArgumentNullException("deviceTelemetryLogic");

}

if (alertsLogic == null)

{

throw new ArgumentNullException("alertsLogic");

}

if (deviceLogic == null)

{

throw new ArgumentNullException("deviceLogic");

}

if (configProvider == null)

{

throw new ArgumentNullException("configProvider");

}

\_deviceTelemetryLogic = deviceTelemetryLogic;

\_alertsLogic = alertsLogic;

\_deviceLogic = deviceLogic;

//Michael

if (bydErrorLogic == null)

{

throw new ArgumentNullException("bydErrorLogic");

}

\_bydErrorLogic = bydErrorLogic;

//End

\_configProvider = configProvider;

}

* + 新增以下Method

private IBYDErrorLogic \_bydErrorLogic = null;

//[HttpGet]

//https://michi-byddemo2-rg.azurewebsites.net/api/v1/telemetry/bydtelemetrydata?start=2016-09-03T00:00:00&end=2016-09-03T00:00:00

[Route("bydyearlyerrorsummary")]

[WebApiRequirePermission(Permission.ViewTelemetry)]

public async Task<HttpResponseMessage> GetYearlyErrorSummaryAsync()

{

var result = await GetServiceResponseAsync(async () =>

{

return await \_bydErrorLogic.GetYearlyErrorsSummaryAsync();

});

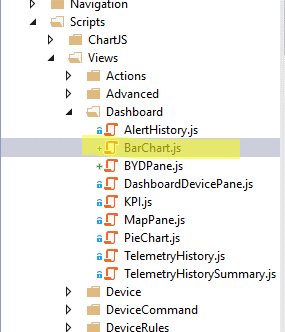
return result;

}

* 在Visual Studio的Web專案中，新增Chart.js NUGET Package



* 在Web專案的Scripts\VIEWS\Dashboard目錄下新增一個BarChart.Js



* + 內容如下

IoTApp.createModule(

'IoTApp.Dashboard.BarChart',

(function () {

var targetControl;

var telemetryDataUrl;

var labels; //["label1", "label2"]

'use strict';

var refreshData = function refreshData() {

if (telemetryDataUrl) {

$.ajax({

cache: true,

//complete: onRequestComplete,

url: telemetryDataUrl

}).done(

function telemetryReadDone(data) {

var monthes = [];

var monthlyData = [];

for (var index = 0; index < data.data.data.length; index++) {

//alert(JSON.stringify(data.data.data[index]));

monthes.push(data.data.data[index].date);

monthlyData.push(data.data.data[index].errorCount);

}

var barChartData = {

labels: monthes,

datasets:

[

{

label: "故障",

fillColor: "rgba(255, 100, 178, 0.2)",

strokeColor: "rgba(255, 100, 178, 0.8)",

highlightFill: "rgba(255, 100, 178, 0.75)",

highlightStroke: "rgba(255, 100, 178, 1)",

data: monthlyData

}

]

};

var chart2 = ($('#barChart'));

new Chart(chart2, {

type: 'bar',

data: barChartData

});

/\*

new Chart(chart2).Bar(barChartData, {

responsive: true,

barValueSpacing: 5,

barDatasetSpacing: 1,

tooltipFillColor: "rgba(0,0,0,0.8)",

multiTooltipTemplate: "<%= datasetLabel %> - <%= value %> "

});

\*/

}

).fail(function () {

alert('failed');

IoTApp.Helpers.Dialog.displayError(resources.unableToRetrieveDeviceTelemetryFromService);

});

}

};

var init = function (settings) {

targetControl = settings.targetControl;

telemetryDataUrl = settings.requestUrl;

};

return {

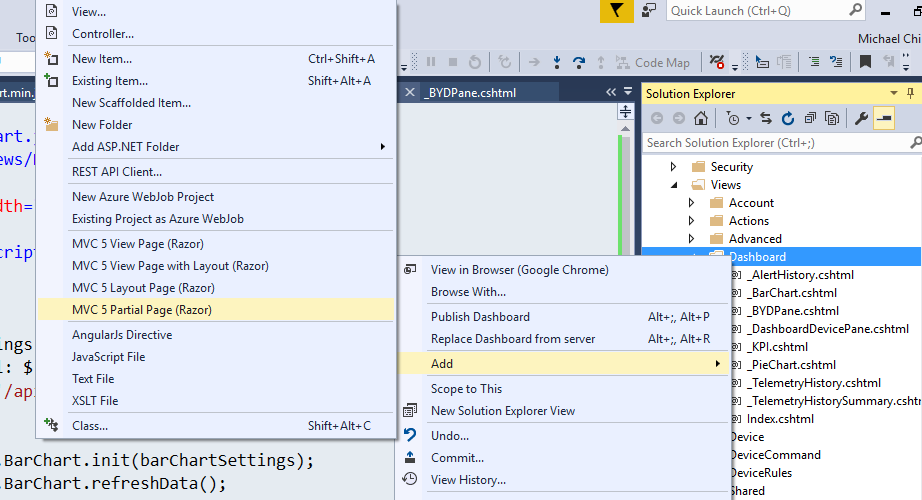
refreshData: refreshData,

init: init

}

}), [jQuery, resources]);

* 在VIEWS\DashBoard目錄下新增一個\_BarChart.CSTML



* 內容如下

@using GlobalResources

<script src="/Scripts/chart.js"></script>

<script src="/Scripts/Views/Dashboard/BarChart.js"></script>

<div>Bar Chat</div>

<canvas id="barChart" width="400" height="400"></canvas>

<script type="text/javascript">

(function () {

'use strict';

var barChartSettings = {

targetControl: $('#barChart'),

requestUrl: '/api/v1/telemetry/bydyearlyerrorsummary'

};

IoTApp.Dashboard.BarChart.init(barChartSettings);

IoTApp.Dashboard.BarChart.refreshData();

})();

</script>

* 修改Index.cshtml

</div>

@{

Html.RenderPartial("\_DashboardDevicePane", Model.DeviceIdsForDropdown);

Html.RenderPartial("\_BarChart");

}

# 即時故障統計圖表



**IOT Suite中原本已經提供了許多可用的Web API包含我們需要的資料；在這裡，我們要展示的是如果使用現有IOT Suite的圖表以及Web API來製作新的圖表。**

* 在此，我們會使用現有的WebApiControllers\TelemetryApiController中的GetLatestAlertHistoryAsync方法，將她稍作修改。
* 修改Web\Models\AlertHistoryDeviceModel.cs中的AlertHistoryDeviceStatus如下

public enum AlertHistoryDeviceStatus

{

AllClear = 0,

Caution,

Critical,

BYD\_Error = 3,

BYD\_Disconnected =4,

BYD\_StandingBy = 5,

BYD\_Shutdown=6,

BYD\_Unknown=7

}

* 修改Web\Models\AlertHistoryDeviceModel.cs新增以下屬性

public AlertHistoryDeviceStatus BYDDeviceStatus

{

get;

set;

}

* 修改IBYDErrorLogic.cs新增以下方法

Task<IEnumerable<BYDError>> GetLastErrorsAsync();

* 修改BYDErrorLogic.cs，新增以下方法Task<IEnumerable<BYDError>> GetLastErrorsAsync()如下：

public async Task<IEnumerable<BYDError>> GetLastErrorsAsync()

{

//假設設備一定會定時送上數據，所以這裡一定有每個設備的最新狀態

//為了方便，這裡我只找前五天的資料，實際上也許需要新增一個新的Method來取得最後一筆狀態碼

var results = await \_errorRepository.GetErrorsAsync(DateTime.UtcNow.AddDays(-5), DateTime.UtcNow.AddMinutes(5));

var grouping = from item in results

group item by item.DeviceId;

var groups = from gp in grouping

select gp.OrderByDescending(item => item.Time).Take(1).FirstOrDefault();

return groups;

}

* 修改WebApiControllers\TelemetryApiController.cs

[HttpGet]

[Route("alertHistory")]

[WebApiRequirePermission(Permission.ViewTelemetry)]

public async Task<HttpResponseMessage> GetLatestAlertHistoryAsync()

{

Func<Task<AlertHistoryResultsModel>> loadHistoryItems =

async () =>

{

// Dates are stored internally as UTC and marked as such.

// When parsed, they'll be made relative to the server's

// time zone. This is only in an issue on servers machines,

// not set to GMT.

DateTime currentTime = DateTime.Now;

var historyItems = new List<AlertHistoryItemModel>();

var deviceModels = new List<AlertHistoryDeviceModel>();

var resultsModel = new AlertHistoryResultsModel();

IEnumerable<AlertHistoryItemModel> data =

await \_alertsLogic.LoadLatestAlertHistoryAsync(

currentTime.Subtract(CautionAlertMaxDelta),

DISPLAYED\_HISTORY\_ITEMS);

if (data != null)

{

historyItems.AddRange(data);

List<dynamic> devices = await LoadAllDevicesAsync();

if (devices != null)

{

//Michael

var deviceStatusBYD = await \_bydErrorLogic.GetLastErrorsAsync();

//End

DeviceListLocationsModel locationsModel = \_deviceLogic.ExtractLocationsData(devices);

if (locationsModel != null)

{

resultsModel.MaxLatitude = locationsModel.MaximumLatitude;

resultsModel.MaxLongitude = locationsModel.MaximumLongitude;

resultsModel.MinLatitude = locationsModel.MinimumLatitude;

resultsModel.MinLongitude = locationsModel.MinimumLongitude;

if (locationsModel.DeviceLocationList != null)

{

Func<string, DateTime?> getStatusTime =

\_deviceTelemetryLogic.ProduceGetLatestDeviceAlertTime(historyItems);

foreach (DeviceLocationModel locationModel in locationsModel.DeviceLocationList)

{

if ((locationModel == null) || string.IsNullOrWhiteSpace(locationModel.DeviceId))

{

continue;

}

var deviceModel = new AlertHistoryDeviceModel()

{

DeviceId = locationModel.DeviceId,

Latitude = locationModel.Latitude,

Longitude = locationModel.Longitude

};

DateTime? lastStatusTime = getStatusTime(locationModel.DeviceId);

if (lastStatusTime.HasValue)

{

TimeSpan deltaTime = currentTime - lastStatusTime.Value;

if (deltaTime < CriticalAlertMaxDelta)

{

deviceModel.Status = AlertHistoryDeviceStatus.Critical;

}

else if (deltaTime < CautionAlertMaxDelta)

{

deviceModel.Status = AlertHistoryDeviceStatus.Caution;

}

}

//Michael

var statusDevice = deviceStatusBYD.Where(b => string.Compare(deviceModel.DeviceId, b.DeviceId, true) == 0).FirstOrDefault();

deviceModel.BYDDeviceStatus = AlertHistoryDeviceStatus.BYD\_Unknown;

if (statusDevice != null) {

switch (statusDevice.StatusCode)

{

case 0:

deviceModel.BYDDeviceStatus = AlertHistoryDeviceStatus.BYD\_Error;

break;

case 1:

deviceModel.BYDDeviceStatus = AlertHistoryDeviceStatus.BYD\_Shutdown;

break;

case 2:

deviceModel.BYDDeviceStatus = AlertHistoryDeviceStatus.BYD\_Disconnected;

break;

case 3:

deviceModel.BYDDeviceStatus = AlertHistoryDeviceStatus.BYD\_StandingBy;

break;

}

}

//End

deviceModels.Add(deviceModel);

}

}

}

}

}

resultsModel.Data = historyItems.Take(DISPLAYED\_HISTORY\_ITEMS).ToList();

resultsModel.Devices = deviceModels;

resultsModel.TotalAlertCount = historyItems.Count;

resultsModel.TotalFilteredCount = historyItems.Count;

return resultsModel;

};

return await GetServiceResponseAsync<AlertHistoryResultsModel>(loadHistoryItems, false);

}

* 新增Web\Scripts\Views\Dashboard\PieChart.js如下：

IoTApp.createModule(

'IoTApp.Dashboard.PieChart',

(function () {

var targetControl;

'use strict';

var refreshData = function refreshData(data) {

//alert(JSON.stringify(data)); //data.devices.BYDDeviceStatus

var bydStatus = { };

bydStatus["0"] = 0;

bydStatus["1"] = 0;

bydStatus["2"] = 0;

bydStatus["3"] = 0;

//alert(data.devices.length);

for(var i = 0; i < data.devices.length; i++){

//alert(JSON.stringify(data.devices[i]));

//alert(data.devices[i].bydDeviceStatus);

bydStatus[data.devices[i].bydDeviceStatus] = bydStatus[data.devices[i].bydDeviceStatus] + 1;

}

//alert(JSON.stringify([bydStatus["0"],bydStatus["1"],bydStatus["2"],bydStatus["3"]]));

var pieData = {

labels: [

"故障",

"關網",

"離網",

"待機",

],

datasets: [

{

data: [bydStatus["0"],bydStatus["1"],bydStatus["2"],bydStatus["3"]],

backgroundColor: [

"#FF6384",

"#36A2EB",

"#FFCE56",

"#000000",

],

hoverBackgroundColor: [

"#FF6384",

"#36A2EB",

"#FFCE56",

"#FFFFFF"

]

}]

};

//Get the context of the canvas element we want to select

var chart2 = $(targetControl);// ($(targetControl));

new Chart(chart2, {

type: 'pie',

data: pieData

});

return;

};

var init = function (settings) {

targetControl = settings.targetControl;

};

return {

refreshData: refreshData,

init: init

}

}), [jQuery, resources]);

* 新增Partial View Web\ Views\Dashboard\\_PieChart.cshtml如下：

@using GlobalResources

<script src="/Scripts/Chart.js"></script>

<script src="/Scripts/Views/Dashboard/PieChart.js"></script>

<div class="dashboard\_device\_map">

<canvas id="pieChart" width="100" height="100"></canvas class="dashboard\_telemetry\_history\_\_telemetry\_history\_graph">

</div>

<script>

(function () {

'use strict';

var pieChartSettings = {

targetControl: ('#pieChart'),

};

IoTApp.Dashboard.PieChart.init(pieChartSettings);

})();

</script>

* 修改Index.cshtl如下：

@using System.Collections.Generic

@using System.Web.Mvc.Html

@using GlobalResources

@using Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Web.Helpers

@model Microsoft.Azure.Devices.Applications.RemoteMonitoring.DeviceAdmin.Web.Models.DashboardModel

@{

ViewBag.Title = "Dashboard";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<script>

'use strict';

var resources = {

active: '@Html.JavaScriptString(Strings.Active)',

alarm: '@Html.JavaScriptString(Strings.Alarm)',

alarmHistory: '@Html.JavaScriptString(Strings.AlarmHistory)',

allClearStatusIcon: '/Content/img/icon\_status\_all\_clear.svg',

cautionStatusIcon: '/Content/img/icon\_status\_caution.svg',

closed: '@Html.JavaScriptString(Strings.Closed)',

criticalStatusIcon: '/Content/img/icon\_status\_critical.svg',

humidity: '@Html.JavaScriptString(Strings.Humidity)',

information: '@Html.JavaScriptString(Strings.Information)',

telemetryGaugeNumericFormat: '@Html.JavaScriptString(Strings.TelemetryGaugeNumericFormat)',

temperature: '@Html.JavaScriptString(Strings.Temperature)',

alertHistoryType: '',

unableToRetrieveAlertsHistoryFromService: '@Html.JavaScriptString(Strings.UnableToRetrieveAlertsHistoryFromService)',

unableToRetrieveDeviceTelemetryFromService: '@Html.JavaScriptString(Strings.UnableToRetrieveDeviceTelemetryFromService)'

};

</script>

<div class="dashboard\_content">

<div class="dashboard\_left\_column">

<div>

@{if (!string.IsNullOrWhiteSpace(Model.MapApiQueryKey))

{

<div id="deviceMap" class="dashboard\_device\_map">

<script charset="UTF-8" type="text/javascript" src="https://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&s=1"></script>

<script type="text/javascript" src="~/Scripts/IoTHelperScripts.js"></script>

<script src="~/Scripts/Views/Dashboard/MapPane.js"></script>

<script>

resources.alertHistoryType = "dashboard\_alert\_history--static\_height";

</script>

</div>

}

else

{

<img class="static\_map" id="staticMap" src="~/Content/img/map\_copyrights.png" alt="Map Img" />

<script>

resources.alertHistoryType = "dashboard\_alert\_history--dynamic\_height";

</script>

}

}

</div>

@{

Html.RenderPartial("\_AlertHistory");

}

</div>

<script type="text/javascript" src="~/Scripts/Views/Dashboard/PieChart.js"></script>

@{

//Html.RenderPartial("\_DashboardDevicePane", Model.DeviceIdsForDropdown);

Html.RenderPartial("\_PieChart");

Html.RenderPartial("\_BarChart");

}

</div>

* 修改AlertHistory.js如下：

var onXhr = function onXhr(e, settings, data) {

if (handleRequestError(settings)) {

if (typeof IoTApp.MapPane === "object" && data) {

IoTApp.MapPane.setDeviceLocationData(

data.minLatitude,

data.minLongitude,

data.maxLatitude,

data.maxLongitude,

data.devices

);

}

//Michael

if (typeof IoTApp.Dashboard.PieChart === "object" && data) {

IoTApp.Dashboard.PieChart.refreshData(

data

);

}

//End

if (refreshMilliseconds) {

if (timerId) {

clearTimeout(timerId);

timerId = null;

}

timerId = setTimeout(reloadGrid, refreshMilliseconds);

}

}

};