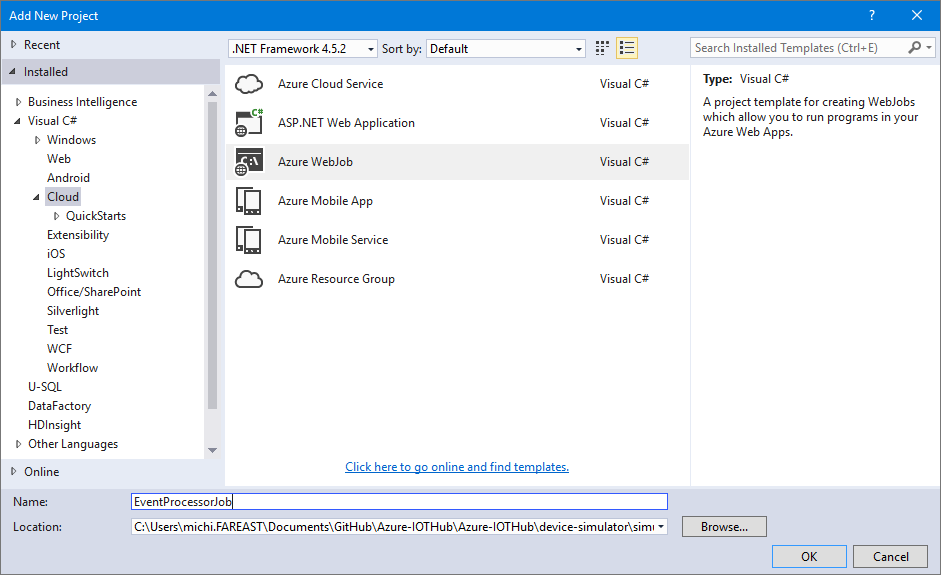
## 環境準備

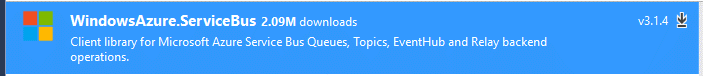
* Visual Studio 2015
* Azure SDK

## 步驟

* 新增一個WebJob專案

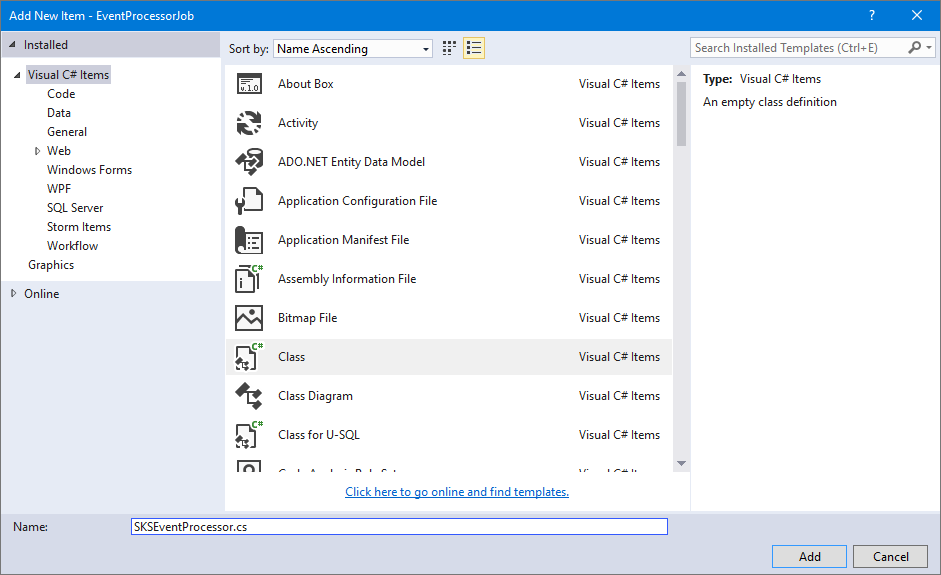


* 加入以下兩個NUGet Package





* 開啟專案，新增一個新的Class名為SKSEventProcessor.cs



* 在SKSEventProcessor.cs加入以下的namespace引用

using System.IO;

using System.Diagnostics;

using System.Security.Cryptography;

using Microsoft.ServiceBus.Messaging;

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Blob;

* 加入以下程式碼

class SKSEventProcessor: IEventProcessor

{

private const int MAX\_BLOCK\_SIZE = 4 \* 1024 \* 1024;

public static string StorageConnectionString;

public static string ServiceBusConnectionString;

private CloudBlobClient blobClient;

private CloudBlobContainer blobContainer;

private QueueClient queueClient;

private long currentBlockInitOffset;

private MemoryStream toAppend = new MemoryStream(MAX\_BLOCK\_SIZE);

private Stopwatch stopwatch;

private TimeSpan MAX\_CHECKPOINT\_TIME = TimeSpan.FromHours(1);

public SKSEventProcessor()

{

var storageAccount = CloudStorageAccount.Parse(StorageConnectionString);

blobClient = storageAccount.CreateCloudBlobClient();

blobContainer = blobClient.GetContainerReference("sksevhblob");

blobContainer.CreateIfNotExists();

queueClient = QueueClient.CreateFromConnectionString(ServiceBusConnectionString, "sksevhqueue");

}

Task IEventProcessor.CloseAsync(PartitionContext context, CloseReason reason)

{

Console.WriteLine("Processor Shutting Down. Partition '{0}', Reason: '{1}'.", context.Lease.PartitionId, reason);

return Task.FromResult<object>(null);

}

Task IEventProcessor.OpenAsync(PartitionContext context)

{

Console.WriteLine("StoreEventProcessor initialized. Partition: '{0}', Offset: '{1}'", context.Lease.PartitionId, context.Lease.Offset);

if (!long.TryParse(context.Lease.Offset, out currentBlockInitOffset))

{

currentBlockInitOffset = 0;

}

stopwatch = new Stopwatch();

stopwatch.Start();

return Task.FromResult<object>(null);

}

async Task IEventProcessor.ProcessEventsAsync(PartitionContext context, IEnumerable<EventData> messages)

{

foreach (EventData eventData in messages)

{

byte[] data = eventData.GetBytes();

var text = Encoding.UTF8.GetString(data);

if (text.IndexOf("\"Type\":1") >= 0)

{

var queueMessage = new BrokeredMessage(new MemoryStream(data));

queueMessage.Properties["message-source"] = "evh";

queueMessage.ContentType = text.GetType().FullName;

await queueClient.SendAsync(queueMessage);

WriteHighlightedMessage(string.Format("\*\*\* Received interactive message: {0}", text));

}

if (toAppend.Length + data.Length > MAX\_BLOCK\_SIZE || stopwatch.Elapsed > MAX\_CHECKPOINT\_TIME)

{

await AppendAndCheckpoint(context);

}

await toAppend.WriteAsync(data, 0, data.Length);

Console.WriteLine(string.Format("Message received. Partition: '{0}', Data: '{1}'",

context.Lease.PartitionId, Encoding.UTF8.GetString(data)));

}

}

private async Task AppendAndCheckpoint(PartitionContext context)

{

var blockIdString = String.Format("startSeq:{0}", currentBlockInitOffset.ToString("0000000000000000000000000"));

var blockId = Convert.ToBase64String(ASCIIEncoding.ASCII.GetBytes(blockIdString));

toAppend.Seek(0, SeekOrigin.Begin);

byte[] md5 = MD5.Create().ComputeHash(toAppend);

toAppend.Seek(0, SeekOrigin.Begin);

var blobName = String.Format("iothubd2c\_{0}", context.Lease.PartitionId);

var currentBlob = blobContainer.GetBlockBlobReference(blobName);

if (await currentBlob.ExistsAsync())

{

await currentBlob.PutBlockAsync(blockId, toAppend, Convert.ToBase64String(md5));

var blockList = await currentBlob.DownloadBlockListAsync();

var newBlockList = new List<string>(blockList.Select(b => b.Name));

if (newBlockList.Count() > 0 && newBlockList.Last() != blockId)

{

newBlockList.Add(blockId);

WriteHighlightedMessage(String.Format("Appending block id: {0} to blob: {1}", blockIdString, currentBlob.Name));

}

else

{

WriteHighlightedMessage(String.Format("Overwriting block id: {0}", blockIdString));

}

await currentBlob.PutBlockListAsync(newBlockList);

}

else

{

await currentBlob.PutBlockAsync(blockId, toAppend, Convert.ToBase64String(md5));

var newBlockList = new List<string>();

newBlockList.Add(blockId);

await currentBlob.PutBlockListAsync(newBlockList);

WriteHighlightedMessage(String.Format("Created new blob", currentBlob.Name));

}

toAppend.Dispose();

toAppend = new MemoryStream(MAX\_BLOCK\_SIZE);

// checkpoint.

await context.CheckpointAsync();

WriteHighlightedMessage(String.Format("Checkpointed partition: {0}", context.Lease.PartitionId));

currentBlockInitOffset = long.Parse(context.Lease.Offset);

stopwatch.Restart();

}

private void WriteHighlightedMessage(string message)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine(message);

Console.ResetColor();

}

}

* 打開Program.cs，加入以下程式碼及namespace；請將iotHubConnectionString、Storage與Service Bus Connection String改為相對應的連線字串

using Microsoft.ServiceBus.Messaging;

//…

static void Main(string[] args)

{

string iotHubConnectionString = "{iot hub connection string}";

string iotHubD2cEndpoint = "messages/events";

SKSEventProcessor.StorageConnectionString = "{storage connection string}";

SKSEventProcessor.ServiceBusConnectionString = "{service bus send connection string}";

string eventProcessorHostName = Guid.NewGuid().ToString();

EventProcessorHost eventProcessorHost = new EventProcessorHost(eventProcessorHostName, iotHubD2cEndpoint, EventHubConsumerGroup.DefaultGroupName, iotHubConnectionString, StoreEventProcessor.StorageConnectionString, "messages-events");

Console.WriteLine("Registering EventProcessor...");

eventProcessorHost.RegisterEventProcessorAsync<SKSEventProcessor>().Wait();

Console.WriteLine("Receiving. Press enter key to stop worker.");

Console.ReadLine();

eventProcessorHost.UnregisterEventProcessorAsync().Wait();

}