Artículo
Yaron Munz · 23 sep, 2022 Lectura de 5 min

Caso de uso de Python Embebido con Azure Service Bus (ASB)

Resumen

Empezamos a usar Azure Service Bus (ASB) como solución de mensajería empresarial hace tres años. La hemos usado para publicar y consumir datos entre muchas aplicaciones de la organización. Como el flujo de datos es complejo, y normalmente se necesitan los datos de una aplicación en muchas otras aplicaciones, el modelo publicador -> múltiples subscriptores resultó muy adecuado. El uso de ASB en la organización es de docenas de millones de mensajes por día, mientras que la plataforma IRIS tiene unos 2-3 millones de mensajes/día.

El problema con ASB

Cuando empezamos con la integración de ASB, encontramos que el protocolo AMQP no tiene la configuración predeterminada para la implementación de IRIS, por lo que estuvimos buscando una solución alternativa para poder comunicar con ASB.

La solución

Desarrollamos un servicio local en Windows (.NET y Swagger) que se encargaba de la comunicación con ASB. Se instaló en la misma máquina que IRIS. Enviamos los mensajes a ASB dirigidos a este servicio local en Windows (usando: localhost y haciendo una API REST) y el propio servicio hacía el resto. Esta solución funcionó bien durante años, pero era muy difícil monitorizar el tráfico, depurar los mensajes "perdidos" y medir el rendimiento general. El hecho de tener este servicio local en Windows como un agente intermedio no resultaba la mejor arquitectura.

Programa de Acceso Preferente a Python Embebido (EAP)

Me pidieron (o me presenté voluntario, no lo recuerdo bien) participar en el Programa de Acceso Preferente a Python Embebido. Fue muy interesante para mí, porque pude poder probar nuevas funcionalidades y dar *feedback* al equipo de desarrollo. También me gustó mucho participar de forma activa e influir en el desarrollo de este producto. En esa fase, empezamos a probar Python Embebido para el ERP y decidimos comprobar si podíamos utilizar Python Embebido para resolver problemas “ reales ” . Nos plantearnos entonces intentar resolver la conectividad directa de IRIS a ASB.

El POC (prueba de concepto)

Empezamos a programar la solución y vimos que usar la librería ASB de Microsoft para Python nos haría nuestra vida (y nuestro trabajo) mucho más fácil. El primer paso fue desarrollar una función que pueda conectar con un *topic* específico en ASB y recibir mensajes. Esto se hizo bastante rápido (1 día de desarrollo) y avanzamos a la fase publicar (enviar mensajes a ASB).

Nos llevó varios días desarrollar toda la cubierta para esas funciones enviar y recibir. Preparamos lo siguiente:

- Un “ área de preparación ” entrante y saliente adecuada, para controlar el flujo de mensajes entrantes/salientes.
- Un mecanismo central para almacenar el tráfico ASB entrante y saliente para poder tener estadísticas

adecuadas y medidas del rendimiento.
- Una página de monitorización CSP para activar/desactivar servicios, mostrar estadísticas y dar alertas sobre cualquier incidencia.

La implementación

Configuración Preliminar

Antes de usar las librerías ASB de Microsoft para Python, hay que instalarlas en un directorio .. \python dedicado. Nosotros elegimos usar ../mge/python/ pero se puede usar cualquier carpeta que se quiera (inicialmente era una carpeta vacía)
Los siguientes comandos hay que ejecutarlos bien con una sesión CMD con privilegios (Windows) o con un usuario con los privilegios suficientes (Linux):
.. \bin \irispip install --target .. \mgr \python asyncio
.. \bin \irispip install --target .. \mgr \python azure.servicebus

Recibir mensajes desde ASB (consumo)

Tenemos una ClassMethod con varios parámetros:

- The *topiId* - El ASB se divide en *topics* de los cuales se consumen los mensajes
- subscriptionName - Nombre de la subscripción Azure
- connectionString - Para poder conectar con el *topic* al que estás suscrito

Ten en cuenta que se está usando [Language = python] para indicar que este ClassMethod está escrito en Python (!)
ClassMethod retrieveASB(topicId As %Integer = 1, topicName As %String = "", subscriptionName As %String = "", connectionString As %String = "", debug As %Integer = 1, deadLetter As %Integer = 0) As %Integer [Language = python]

Para usar la librería ASB, primero necesitamos importar las librerías ServiceBusClient y ServiceBusSubQueue:
```
from azure.servicebus import ServiceBusClient,ServiceBusSubQueue
```

para poder interactuar con IRIS (por ejemplo, ejecutar código) también necesitamos:
```
import iris
```

En este punto, podemos usar las librerías ASB:
```
with ServiceBusClient.fromconnectionstring(connectionString) as client:
  with client.getsubscriptionreceiver(topicName, subscriptionName, maxwaittime=1, subqueue=subQueue) as receiver:

  for msg in receiver:
```

En este punto, tenemos un objeto Python "msg" (stream) donde podemos pasarlo (como un stream, por supuesto) a IRIS y almacenarlo directamente en la base de datos:
```
result = iris.cls("anyIRIS.Class").StoreFrom Python(stream)
```

Enviar mensajes a ASB (publicación)

Tenemos un ClassMethod con varios parámetros:

- topicName - El Topic en el que queremos publicar (aquí tenemos que pasar el nombre, no el id)
- connectionString - Para poder conectar con el topic al que se está suscrito

- JSONmessage - el mensaje que queremos enviar (publicar)

Ten en cuenta que se está usando [Language = python] para indicar que este ClassMethod está escrito en Python (!)
ClassMethod publishASB(topicName As %String = "", connectionString As %String = "", JSONmessage As %String = "") As %Status [Language = python]

Para usar la librería ASB, primero necesitamos importar las librerías ServiceBusClient y ServiceBusMessage:
from azure.servicebus import ServiceBusClient, ServiceBusMessage

Usar las librerías ASB es muy sencillo:

```
try:
  result=1

  with ServiceBusClient.fromconnectionstring(connectionString) as client:

  with client.getqueuesender(topicName) as sender:

  singlemessage = ServiceBusMessage(JSONmessage)

  sender.sendmessages(singlemessage)

  except Exception as e:

  print(e)

  result=0

  return result
```

Beneficios de usar la conectividad ASB directa

- Mucho más rápida que usar la antigua alternativa del servicio local en Windows
- Fácil de monitorizar, recoger estadísticas, resolver incidencias
- Retirar el agente intermedio (servicio local en Windows) reduce un potencial punto de fallo
- Posibilidad de gestionar mensajes perdidos para cualquier *topic* automáticamente y hacer que ASB re-envíe esos mensajes a los subscriptores del *topic*

Agradecimientos

Me gustaría dar las gracias a @David Satorres (Senior Developer para IRIS) por su enorme contribución en el diseño, programación y pruebas. Sin su ayuda este proyecto no hubiera sido posible.

#Azure #Embedded Python #Mejores prácticas #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/caso-de-uso-de-python-embebido-con-azure-service-bus-asb