記事
Toshihiko Minamoto · 2021年12月15日 2m read

Ensemble とFile Outbound Adapter - ちょっとしたヒント

あるお客様の問題から、この短い記事を書くことにしました。 お客様はEnsembleを使用して、多数のシステムを統合しています。一部のシステムではプレーンファイルのみが使用されています。

そのため、ターゲットファイルへの書き込みには、自然とFile Outbound Adapter を選択しました。 数年もの間すべてが順調に稼働していましたが、最近になって、ファイルに書き込まれるデータが数十メガバイトという大きなサイズに達するようになり問題が出てきました。オペレーションが完了するまでに約30分かかるようになり、プロセス内の後続の処理を待たせなければならないタイミングの問題が発生し始めたのです。当然、連携先のシステムはそれほど長く待つことを良しとしません。

お客様のコードは、以下の疑似コードのようなものでした。

```
 set tResultSet=SQLStatement.Execute()

 // compose header based on resultset columns Describe()

 set tSC= ..Adapter.PutLine(file,header)

while tResultSet.%Next() {

 set line=... compose line of the resultset row data

  set tSC=..Adapter.PutLine(file,line)

}
```

アダプターのPutLine() メソッドのソースコードを見ると、このメソッドがファイルを開き、行を書き込んで、ファイルを閉じていることがわかりました。 しかも、行ごとにです！！！

そこで、より良いアダプターAPIを提供できないか確認したところ、PutStream()
メソッドというのを見つけました。
大当たりです！これが正しいメソッドで、ストリームを毎回開かずに、ただ書き込むメソッドです。

コードを以下のように書き換えました。

```
set tTmpStream=##class(%FileCharacterStream).%New()

set tResultSet=SQLStatement.Execute()

 // compose header based on resultset columns Describe()

 set tSC= ..tTmpStream.WriteLine(header)

while tResultSet.%Next() {

 set line=... compose line of the resultset row data
```

```
 set tSC=tTmpStream.WriteLine(line)

}

set tSC=..Adapter.PutStream(file,tTmpStream)
```

お客様は上記の変更したコードを使用して、オペレーションを再実行しました。
これで、オペレーションにかかる時間は数秒になりました！

結果: ファイルを処理する場合、PutLine() が最初の（そして最も簡単な）選択肢かもしれませんが、本番環境では必ずしも良い方法とは言えません。特に大きなファイルを取り扱う場合は尚更です。

今後の開発プロジェクトのお役に立てられれば幸いです。

Dan Kutac

#ヒントとコツ #ビジネスオペレーション #Ensemble

ソースURL:https://jp.community.intersystems.com/post/ensemble-%E3%81%A8file-outbound-adapter-%E3%81%A1%E3%82%87%E3%81%A3%E3%81%A8%E3%81%97%E3%81%9F%E3%83%92%E3%83%B3%E3%83%88