SECURE iNET SUITE

Secure iNetSuite for .NET 4.0J の新仕様について

グレープシティ 株式会社

2013 年 8 月 初版

メール送受信とファイル転送機能を実現する通信コンポーネント「Secure iNet Suite」の通信モードの仕様が新しくなりました。本資料では従来のバージョンとの違いとメリットをコードを使って詳しく解説します。

はじめに

2013 年 9 月発売の Secure FTP for .NET 4.0J と Secure Mail for .NET 4.0J では、非同期処理と同期処理の 2 つの通信モードの実装方法を同じロジックで行えるよう直前のバージョンである Secure FTP 2.0J と Secure Mail 2.0J から全面的に仕様変更を行いました。その結果として両バージョン間での互換性が失われることになりましたが、より多くの利点が生まれています。本資料では主な仕様変更の内容と変更理由から、新バージョンでのメリットについて解説していきます。

2013 年 8 月

Contents

1. 非同期処理の実装方法の変更

通信処理を実現する方法として、同期モード（ブロッキングモード）と非同期モード（非ブロッキングモード）があります。同期モードでは、処理はすべてメインスレッドで実行されるため、通信処理中は他のすべての処理がブロックされます。非同期モードでは、処理はバックグラウンドで実行されるため、処理中もメインスレッド上での他の処理、例えばユーザーによるデータ入力などの別の作業を継続することができます。

通信処理のみを行うアプリケーションであれば、通信中に他の処理がブロックされても問題はないかもしれませんが、アプリケーションの機能の一部として FTP や Mail などの通信機能を組み込む場合、通信中に他の作業を継続できないと作業効率が悪くなり、実業務では使いにくい不便なシステムとなります。このようなアプリケーションでは非同期モードでの通信処理が求められます。

この章では新バージョン（4.0J）と従来のバージョン（2.0J）での非同期処理の違いについて同じ処理を行うコードを用い、比較しながら説明します。

処理内容は、次の手順で FTP サーバーから JPEG ファイルをダウンロードするものとします。

1. FTP サーバーにログインし、カレントディレクトリを取得する
2. 「TEST」ディレクトリに移動する
3. 「TEST」ディレクトリ内にある JPEG ファイルをダウンロードする
4. FTP サーバーからログアウトする

従来のバージョンでの実装方法

同期モードで作成した場合のコード

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Button1.Enabled = False

    With Ftp1
        .Server = tbxServer.Text
        .Username = tbxUserName.Text
        .Password = tbxPassWord.Text

        '(1)FTPサーバーにログインし、カレントディレクトリを取得する
        .GetDirectory()

        '(2)「TEST」ディレクトリに移動する
        .Invoke(Dart.PowerTCP.SecureFtp.FtpCommand.ChangeDir, "TEST")

        '(3)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
        .Get("", "*.jpg", "C:¥Temp¥FTPTest2", False, False)

        '(4) FTPサーバーからログアウトする
        .Close()
    End With

    Button1.Enabled = True
End Sub
```

非同期モードで作成した場合のコード

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Button1.Enabled = False

        With Ftp1
            .Server = tbxServer.Text
            .Username = tbxUserName.Text
            .Password = tbxPassWord.Text

            '(1)FTPサーバーにログインし、カレントディレクトリを取得する
            .BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.PrintDir, "GetDirectry")
        End With
    End Sub

    Private Sub Ftp1_EndInvoke(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.InvokeEventArgs)
Handles Ftp1.EndInvoke
        If e.State = "GetDirectry" Then
            '(2)「TEST」ディレクトリに移動する
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.ChangeDir, "TEST", "ChangeDirectry")
        ElseIf e.State = "ChangeDirectry" Then
            '(3)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
            Ftp1.BeginGet("", "*.jpg", "C:¥Temp¥FTPTest2", False, False, "GetFile")
        ElseIf e.State = "Quit" Then
            Button1.Enabled = True
        End If
    End Sub

    Private Sub Ftp1_EndGet(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.FileEventArgs) Handles
Ftp1.EndGet
        If e.State = "GetFile" Then
            '(4) FTPサーバーからログアウトする
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.Quit, "Quit")
        End If
    End Sub
```

従来の非同期処理コードの問題点

1. **同期モードと非同期モードでは使用するメソッドが異なるため、同じ処理を実行するためにも別のコーディング方法を覚えなければならない**
   例えば、同期モードの Get()に相当する非同期のメソッドは BeginGet()ですが、GetDirectry()や Close()には、BeginGetDirectry()や BeginClose()メソッドが無く、BeginInvoke()メソッドを使う必要があります。
   また、BeginGet()メソッドを使用した場合、次の処理コードは EndGet イベントに記述しますが、

BeginInvoke()メソッドを使用した場合は、EndInvoke イベントに次の処理コードを記述する必要があります。

2. **処理の流れが把握しにくくメンテナンス性が悪い**

上記のように、非同期モードでも処理を実現するための方法が統一されていないため、実装方法が分かりにくく、コードの保守性や拡張性についてはあまり良いとは言えません。
例えば、「TEST」ディレクトリに移動し、ファイルをダウンロードする前にディレクトリ内のファイルの List を取得する処理を追加したい場合、次のように変更する必要があります。

同期モードの場合

```
'(2)「TEST」ディレクトリに移動する
.Invoke(Dart.PowerTCP.SecureFtp.FtpCommand.ChangeDir, "TEST")

'(3)Listメソッドを実行するコードを挿入するだけで済む
Dim fList As Dart.PowerTCP.SecureFtp.Listing = .List("*.*", True)

'(4)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
.Get("", "*.jpg", "C:¥Temp¥FTPTest2", False, False)
```

非同期モードの場合

```
    Private Sub Ftp1_EndInvoke(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.InvokeEventArgs)
Handles Ftp1.EndInvoke
        If e.State = "GetDirectry" Then
            '(2)「TEST」ディレクトリに移動する
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.ChangeDir, "TEST", "ChangeDirectry")
        ElseIf e.State = "ChangeDirectry" Then
            '(3)「TEST」ディレクトリのファイルリストを取得する
            Ftp1.BeginList("*.*", True, "GetList")
            'Ftp1.BeginGet("", "*.jpg", "C:¥Temp¥FTPTest2", False, False, "GetFile")
        ElseIf e.State = "Quit" Then
            Button1.Enabled = True
        End If
    End Sub

    Private Sub Ftp1_EndList(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.ListEventArgs) Handles
Ftp1.EndList
        If e.State = "GetList" Then
            '(4)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
            Ftp1.BeginGet("", "*.jpg", "C:¥Temp¥FTPTest2", False, False, "GetFile")
        End If
    End Sub

    Private Sub Ftp1_EndGet(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.FileEventArgs) Handles
Ftp1.EndGet
        If e.State = "GetFile" Then
            '(5) FTPサーバーからログアウトする
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.Quit, "Quit")
        End If
    End Sub
```

非同期モードではファイルリストを取得するために BeginList()メソッドを使用し、次の処理を EndList イベントに記述するため、BeginGet()メソッドを記述する場所を EndInvoke イベントから EndList イベントに変更しなければなりません。このようにアプリケーションのちょっとした仕様変更でも、イベントが増えたりコードを記述する場所が変わったりと、コードが複雑になるためバグが混入するリスクが高まります。

では、同じ処理を新バージョンで作成した場合はどのようになるのかを次の章で説明します。

新バージョンでの実装方法

同期モードで作成した場合のコード

```
Imports Dart.Ftp

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Dim session As New FtpSession
        session.RemoteEndPoint.HostNameOrAddress = tbxServer.Text
        session.Username = tbxUserName.Text
        session.Password = tbxPassWord.Text

        'ファイル取得の処理を同期モードで実行します
        GetFiles(session)
    End Sub

    Private Sub GetFiles(ByVal session As FtpSession)
        With Ftp1
            .Marshal("Start", False)
            .Session = TryCast(session, FtpSession)
            .Connect()
            .Authenticate()

            '(1)カレントディレクトリを取得する
            Ftp1.GetDirectory()

            '(2)「TEST」ディレクトリに移動する
            Ftp1.SetDirectory("TEST")

            '(3)「TEST」ディレクトリのファイルリストを取得する
            Dim filesToGet As List(Of ListEntry) = Ftp1.List("", "*.jpg", ListType.Full)

            '(4)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
            Dim result As List(Of Dart.Ftp.CopyResult) = Ftp1.Get(filesToGet, "/TEST", "C:¥Temp¥FTPTest2",
Synchronize.Off)

            '(5) FTPサーバーからログアウトする
            .Close()
            .Marshal("Finish", True)
        End With
    End Sub

    Private Sub Ftp1_UserState(sender As System.Object, e As Dart.Ftp.UserStateEventArgs) Handles Ftp1.UserState
        If e.Message = "Start" Then
            Button1.Enabled = False
        Else
            Button1.Enabled = True
        End If
    End Sub
```

非同期モードで作成した場合のコード

同じ処理を非同期モードで作成した場合のコードは次のようになります。

```
Imports Dart.Ftp

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Dim session As New FtpSession
        session.RemoteEndPoint.HostNameOrAddress = tbxServer.Text
        session.Username = tbxUserName.Text
        session.Password = tbxPassWord.Text

        'ファイル取得の一連の処理を非同期モードで実行します
        Ftp1.Start(AddressOf GetFiles, session)
    End Sub

    Private Sub GetFiles(ByVal session As FtpSession)
        With Ftp1
            .Marshal("Start", False)

            .Session = TryCast(session, FtpSession)

            .Connect()
            .Authenticate()

            '同期モードと同様のため、以下省略します。
        End With
    End Sub

    Private Sub Ftp1_UserState(sender As System.Object, e As Dart.Ftp.UserStateEventArgs) Handles Ftp1.UserState
        If e.Message = "Start" Then
            Button1.Enabled = False
        Else
            Button1.Enabled = True
        End If
    End Sub
```

このように新バージョンでは同期モードと非同期モードでのコードの違いは下記の点のみです。

```
    'ファイル取得の処理を同期モードで実行します
        GetFiles(session)

    'ファイル取得の一連の処理を非同期モードで実行します
        Ftp1.Start(AddressOf GetFiles, session)
```

新バージョンの非同期処理コードの利点

1. **ファイルを取得する処理内容は同期モードでも非同期モードでも同じなので、ビジネスロジックを一か所（この例では GetFiles）にまとめることができる**
   GetFiles をメインスレッドから直接呼び出した場合は同期モードとなり、Start メソッドを介して呼び出した場合は、一連の処理はバックグラウンドのスレッドで実行される非同期モードになります。

   通信処理のための、List や Get メソッドなどはすべて同期モードでも非同期モードでも同じメソッドを使用できるため学習効率も高く、可読性やメンテナンス性、拡張性の高いコードを記述することができる

2. **バックグラウンドスレッドからのメインスレッドの呼び出しを必要最小限とし、実行効率性の高い非同期処理を実現できる**
   従来の非同期処理では、実際に通信処理を実行している間だけが非同期で実行されており、一つの処理が完了するたびにメインスレッドに処理を戻していました。新バージョンでは、通信中の処理だけでなく、ログインからログアウトまでの一連の処理をすべてまとめて非同期のスレッドで実行することができます。処理の完了時や途中でエラーが発生した場合など、必要な場合だけ任意にメインスレッドに処理を戻すことができるので、マルチスレッドによる非同期処理のメリットやパフォーマンスを最大限に利用することができます。

実は従来の非同期処理は、Visual Basic 6.0（以下 VB6）時代の設計を引き継いだものでした。
VB6 ではマルチスレッドプログラムを作成することができなかったため、上記で紹介した新バージョンのようなバックグラウンドのスレッドを使用した非同期処理をコーディングすることができませんでした。
このため、VB6 用に提供していた「iNetTransfer※1」や「iNetMail※1」では、非同期用のメソッドとイベントを使用した方法で非同期処理を実現していました。

.NET Framework でマルチスレッドプログラミングが可能になりましたが、.NET Framework 1.x では Thread クラスや Delegate、Invoke などを扱う必要があり、実現方法はまだ複雑なものでした。また .NET 用にリリースされた「iNet FTP for .NET」や「iNet Mail for .NET」では VB6 から .NET への移行をサポートするために従来の非同期処理方法が継承されました。しかし .NET Framework も進化し、BackgroundWorker を使ったシンプルなマルチスレッド処理や、最新の.NET Framework 4.5 では Async／Await キーワードを使用して、さらにスマートなマルチスレッド処理を簡単に実装できるようになりました。

このような時代の変化や技術の進化に伴い、新バージョンでは実用的なアプリケーションに求められる非同期処理の実現方法を見直し、.NET Framework の新しい技術を取り入れた設計に変更しました。
※1：ActiveX 製品。現在は販売を終了しています。

2. MVC(Model-View-Controller)設計パターンへの対応

メンテナンス性や拡張性に優れ、品質の高いプログラムを開発する手法の一つとして、MVC（Model-View-Controller）設計パターンがあります。これはアプリケーションを次の３つの要素に分離してプログラムの見通しを良くし、UI やビジネスロジックの変更などのアプリケーションの仕様変更や機能拡張にも柔軟かつ迅速に対応できるようにする設計手法です。

1. Model（モデル）
   データの処理を行うビジネスロジック層です。
2. View（ビュー）
   データの処理結果を表示したり、ユーザーが入力操作を行うための UI 層です。
3. Controller（コントローラ）
   View でのユーザーの入力を受け取り、それに応じて適切な Model を呼び出し、アプリケーションの処理を制御します。

実用性の高いアプリケーションのための非同期処理の必要性や重要性は先に説明しましたが、従来の非同期処理方法では、この MVC 設計パターンに沿ったプログラムは作成できませんでした。ここで、従来の非同期モードで作成したコードをもう一度確認します。

従来バージョンの非同期モードで作成した場合のコード

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Button1.Enabled = False

        With Ftp1
            .Server = tbxServer.Text
            .Username = tbxUserName.Text
            .Password = tbxPassWord.Text

            '(1)FTPサーバーにログインし、カレントディレクトリを取得する
            .BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.PrintDir, "GetDirectry")
        End With
    End Sub

    Private Sub Ftp1_EndInvoke(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.InvokeEventArgs) Handles
Ftp1.EndInvoke
        If e.State = "GetDirectry" Then
            '(2)「TEST」ディレクトリに移動する
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.ChangeDir, "TEST", "ChangeDirectry")
        ElseIf e.State = "ChangeDirectry" Then
            '(3)「TEST」ディレクトリにあるJPEGファイルをダウンロードする
            Ftp1.BeginGet("", "*.jpg", "C:¥Temp¥FTPTest2", False, False, "GetFile")
        ElseIf e.State = "Quit" Then
            Button1.Enabled = True
        End If
    End Sub

    Private Sub Ftp1_EndGet(ByVal sender As Object, ByVal e As Dart.PowerTCP.SecureFtp.FileEventArgs) Handles
Ftp1.EndGet
        If e.State = "GetFile" Then
            '(4) FTPサーバーからログアウトする
            Ftp1.BeginInvoke(Dart.PowerTCP.SecureFtp.FtpCommand.Quit, "Quit")
        End If
    End Sub
```

このようにイベントドリブンなコードとなるため、Model 部と Controller 部を分離することができません。
これに対し、新バージョンの場合は次のようになります。

新バージョンの非同期モードで作成した場合のコード

```
Imports Dart.Ftp

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Dim session As New FtpSession
        session.RemoteEndPoint.HostNameOrAddress = tbxServer.Text
        session.Username = tbxUserName.Text
        session.Password = tbxPassWord.Text

        'ファイル取得の一連の処理を非同期モードで実行します
        Ftp1.Start(AddressOf GetFiles, session)
    End Sub

    Private Sub GetFiles(ByVal session As FtpSession)
        With Ftp1
            '省略しました
        End With
    End Sub

    Private Sub Ftp1_UserState(sender As System.Object, e As Dart.Ftp.UserStateEventArgs) Handles Ftp1.UserState
        If e.Message = "Start" Then
            Button1.Enabled = False
        Else
            Button1.Enabled = True
        End If
    End Sub
```

このように、ビジネスロジックをまとめた「GetFiles」プロシージャが Model 部で、ユーザーのボタン Click という入力に応答してその処理を呼び出している「Button1_Click」が Controller 部となり、Model と Controller を分離することができるため、MVC 設計パターンに沿ったコードを作成することができます。

まとめ

以上のように、新バージョンでは.NET Framework の新しい技術を取り入れ、作業効率の高いアプリケーションの開発に必要な非同期処理をより実用的なスタイルに進化させました。また、非同期処理を実装する場合でも、MVC 設計パターンに沿った開発を可能とし、可読性の高いプログラムコードによりバグの発生リスクを抑え、高品質で保守性や拡張性にも優れたアプリケーションの開発をさらに強力にサポートするために設計を一から見直し、仕様の変更を行いました。新しい仕様により、同期／非同期モードで一貫性のあるプログラミングモデルを提供します。さらに使いやすくなった新バージョンの開発生産性の高さを実感して頂き、セキュアでパフォーマンス性の高いアプリケーションの開発にお役立て頂ければ幸いです。

本製品に関するお問い合わせはツール事業部 営業部までお気軽にご相談ください。

グレープシティ株式会社 ツール事業部 営業部
TEL ：048-222-3001
E-メール ：sales@grapecity.com