記事
Toshihiko Minamoto · 2022年2月22日 14m read

InterSystems IRIS で IMAP クライアントを実装する - パート II

最初のパートでは、IMAP プロトコルコマンドについて簡単に説明しました。このパートでは、IRIS を使用してこれらのコマンドを実装し、独自の IMAP クライアントを作成してみましょう！

IRIS Email フレームワーク

IRIS プラットフォームには電子メールを操作するためのデフォルトのインターフェースとクラスがあります。
これらのアーティファクトは元々 POP3 実装用に設計されていますが、 これらのインターフェースとクラスを IMAP クライアントの実装に使用して拡張できないということではありません。
それでは、このことについて説明しましょう。

%Net.FetchMailProtocol: メールを取得するための基本クラスです。 IMAP クライアントはこれを拡張しています。
%Net.MailMessage: MIME メッセージです。 %Net.MailMessagePart. を拡張します。
%Net.MailMessagePart: マルチパートメッセージの MIME メッセージをカプセル化します。このクラスにはそれ自体の配列があり、メッセージのサブパートをツリーで表現できるようにします。
%Net.MIMEReader: このユーティリティは、メッセージの MIME コンテンツを解析するメソッドで、%Net.MIMEPart インスタンスを生成します。
%Net.MIMEPart: メッセージの MIME パートをカプセル化し、それらに関する情報を取得するためのメソッドを提供します。

IMAP クライアントの実装

このセクションでは、IMAP クライアント、インバウンドアダプタ、および単純なプロダクションに関する実装内容を紹介します。
スペースを節約するために、ほとんどの実装メソッドは説明していないことに注意してください。
代わりに、それぞれの実装の完全な詳細へのリンクを付けています。
完全なソースコードは、GitHub から入手できます。

基本的な IMAP クライアントの作成

前のパートで説明したように、IMAP は TCP を介すプレーンテキストベースのプロトコルです。
つまり、そのようなプロトコル向けにクライアントを実装するベースコードは TCP クライアントということになります。

IRIS プラットフォームには、OPEN、USE、READ、WRITE、および CLOSE という I/O 操作を実行するための標準の ObjectScript コマンドが提供されています。

以下に、MS Outlook サーバーに接続し、ログインしてからログアウトする方法例を簡単に示します。

```
ClassMethod SimpleTest()
{
    // connection configuration
    SET dev = "|TCP|993"
    SET host = "outlook.office365.com"
    SET port = "993"
    SET mode = "C"
    SET sslConfig = "ISC.FeatureTracker.SSL.Config"
    SET timeout = 30
    // connection to MS Outlook IMAP server
    OPEN dev:(host:port:mode:/TLS=sslConfig):timeout
    THROW:('$TEST) ##class(%Exception.General).%New("Sorry, can't connect...")
    USE dev
    READ resp($INCREMENT(resp)):timeout
    WRITE "TAG1 LOGIN user@outlook.com password", !
    READ resp($INCREMENT(resp)):timeout
    WRITE "TAG2 LOGOUT", !
    READ resp($INCREMENT(resp)):timeout
    CLOSE dev
    // come back to default device (terminal) and prints responses
    USE 0
    ZWRITE resp
}
```

この出力は以下のようになります。

```
USER>d ##class(dc.Demo.Test).SimpleTest()
resp=3
resp(1)="* OK The Microsoft Exchange IMAP4 service is ready. [QwBQ..AA==]"_$c(13,10)
resp(2)="TAG1 OK LOGIN completed."_$c(13,10)
resp(3)="* BYE Microsoft Exchange Server IMAP4 server signing off."_$c(13,10)_"TAG2 O
K LOGOUT completed."_$c(13,10)
```

このコードにはいくつかのハイライトがあります。

mode 変数を C に設定します。これはキャリッジリターンモードです。 IMAP の場合、この設定は必須です。
フラグ /TLS によって安全な通信レイヤー（SSL）を確立します。 このフラグ値を有効な SSL IRIS
接続に設定する必要があります。
OPEN コマンドによって、接続を開始します。
特殊なブール変数の $TEST
は、タイムアウ
ト付きのコマンドが成功した場
合またはタイムアウトが期限切れになった場合に 1 を返します。 この例では、OPEN コマンドが 30
秒を超えると、コードは例外をスローします。
接続が確立したら、コマンド USE によって TCP
デバイスが所有され、すべての READ と WRITE
コマンドがこのデバイスにリダイレクトされるようになります。
WRITE コマンドが IMAP サーバーにコマンドを発行し、READ コマンドがその出力を取得します。
接続を終了するために、CLOSE コマンドを使用します。
デバイスを所有すると、READ と WRITE へのすべての呼び出しは、dev
変数に指定されているデバイスで、USE dev コマンドを使用した後に実行されます。
ターミナルに戻ってもう一度書き込む場合は、先に USE 0 コマンドを発行する必要があります。

それぞれの READ コマンドには、サーバーの応答を格納するバッファがありますが、容量が制限されています。応答のサイズがこの制限を上回る場合は、別の READ コマンドを発行して全応答を読み取る必要があります。 もちろん、バッファサイズを増やすこともできますが、制限を超えるような状況に対処する準備を整えておく方が適しています。

前のパートで説明したように、IMAP の各コマンドにはタグが必要です。
このタグは、コードが完全な応答を取得したか、別の READ
コマンドを発行する必要があるかどうかをチェックする上で役立ちます。
この場合は、ReadResponse
メソッドを実装して、コードが確実にメッセージ全体を読み取るようにしています。

IMAP の %Net.FetchMailProtocol インターフェースの実装

%Net.FetchMailProtocol 抽象クラスは、IRIS プラットフォームでの電子メールの取得を抽象化します。
実装するのは以下のメソッドです。

Connect: IMAP サーバーへの接続を確立し、ユーザーをログインさせます。
GetMailBoxStatus: メールボックスのサイズとその中のメッセージの件数を取得します。
GetSizeOfMessages: メッセージ番号で識別される 1 件またはすべてのメッセージのサイズを取得します。
GetMessageUIDArray: 受信トレイ内の 1 件またはすべてのメッセージ UID を配列で取得します。
GetMessageUID: メッセージ番号に対応する UID を取得します。
Fetch:
メッセージ番号
で、メッセージのコンテンツ（
またはマルチパートコンテンツ）を取得します。 %Net.MailMessage
オブジェクトにカプセル化されたメッセージコンテンツを取得します。
FetchFromStream: これは Fetch と同じですが、IMAP
サーバーを呼び出す代わりに、%BinaryStream オブジェクトにカプセル化された EML
メッセージコンテンツからコンテンツを取得します。
FetchMessage: これは Fetch と同じですが、ByRef 変数で特定のメッセージヘッダーを返します。
FetchMessageInfo: メッセージヘッダーとメッセージのテキストのみを取得します。
DeleteMessage: 削除用の配列にメッセージを追加します。
RollbackDeletes: 削除用の配列をクリーンアップします。
QuitAndCommit: 削除用の配列のすべてのメッセージを削除し、IMAP サーバーから切断します。
QuitAndRollback: 削除用の配列をクリーンアップし、IMAP サーバーから切断します。
Ping: セッションをアクティブに維持するように IMAP に ping を送信します。
まず、インターフェースを実装するために、dc.Demo.IMAP という新しいクラスを作成します。
このクラスはいくつかのプロパティを継承します。IMAP
サーバーへの接続を確立するように設定する必要があります。

また、dc.Demo.IMAPHelper というヘルパークラスも作成します。 このクラスは、IMAP 応答のメソッドを解析し、マルチパートメッセージのすべてもパートを取得し、コマンドを送信して応答全体が読み取られるようにするメソッドなどの周辺機能を格納します。

最初に実装するメソッドは、Connect メソッドです。
このメソッドは、クラスプロパティにカプセル化された構成を使用して、IMAP サーバーへの接続を確立します。
ログインも発行します。 このメソッドは、IRIS プラットフォームの OPEN コマンドを使用して IMAP
サーバーへの接続を確立し、IMAP コマンドの LOGIN を使用してサーバーへの認証を行います。

次に実装するメソッドは GetMailBoxStatus です。
このメソッドは、SELECT
コマンドを使用してメールボックスを選択し、メールボックス内のメッセージ件数などの追加情報を集めます。

IMAP には、すべてのメッセージのサイズを取得するためにすぐに使用できるコマンドがありません。

もちろん、すべてのメッセージを反復処理して、サイズの合計を計算することは可能ですが、
この方法では、処理速度の低下の問題が発生する可能性があります。
そこで、この実装では、すべてのメッセージのサイズは取得しません。

次のメソッドは GetSizeOfMessages です。 このメソッドは、受信トレイ内の 1
件以上のメッセージのサイズを取得します。
メッセージ番号が定義されていない場合、GetMailBoxStatus メソッドで説明した IMAP
の制限と同じ理由で例外がスローされます。 ここでは、IMAP コマンドの FETCH <messagenumber>
(RFC822.SIZE) を使用して、番号でメッセージのサイズを取得します。

次は、GetMessageUIDArray メソッドで、これは IMAP コマンドの SELECT と UID SEARCH [ALL ¦
<messagenumber>] を使用して応答を解析し、UID 配列を取得します。

そして GetMessageUID メソッドを実装します。 このメソッドは、定義されたメッセージ番号の UID
を取得して、GetMessageUIDArray メソッドと同じロジックを使用します。

その後に、Fetch メソッドを実装します。 これは、IMAP コマンドの SELECT と FETCH <messagenumber>
BODY を使用して、MIME format でエンコードされているメッセージのコンテンツを取得します。
幸いなことに、IRIS プラットフォームには、MIME
コンテンツのリーダーである %Net.MIMEReader クラスがあります。
このクラスは、ストリーム内の
メッセージを取得して、解析済みのメッセージを %Net.MIMEPart オブジェクトで返します。

このメソッドは、MIME
コンテンツを取得したら、%Net.MailMessage オブジェクトを作成し、%Net.MIMEPart
オブジェクトのデータを挿入してそれを返します。

MIME
コンテンツは、dc.Demo.IMAPHelper クラスの GetMailMessageParts メソッドを介して
%Net.MailMessagePart オブジェクトにマッピングする %Net.MIMEPart
オブジェクトにカプセル化されています。

次のメソッドは、FetchFromStream です。 このメソッドは、EML
メッセージと共にストリームオブジェクトを取得し、それを %Net.MailMessage オブジェクトに変換します。
このメソッドはサーバーからコンテンツを取得しません。

次は、Fetch メソッドの特殊ケースである FetchMessage と FetchMessageInfo メソッドです。

DeleteMessage
メソッドは、メッセージを削除するようにマー
クするのに対し、RollbackDeletes
メソッドは、削除用にマークされたメッセージの配列をクリーンアップするだけです。

次は、QuitAndCommit メソッドです。 これは、IMAP
サーバーから切断し、メッセージを削除するための CommitMarkedAsDeleted メソッドを呼び出します。

QuitAndRollback メソッドは、IMAP
サーバーから切断して、削除用にマークされたメッセージの配列をクリーンアップするだけです。

最後のメソッドは Ping で、これは NOOP コマンドを発行して IMAP セッションをアクティブに維持します。

IMAP のインバウンドアダプタの実装

IRIS プラットフォームにおけるインバウンドメールの基本クラスは EnsLib.EMail.InboundAdapter です。
このインバウンドアダプタには、以下の構成が必要です。

メールサーバーのホストアドレス
メールサーバーのポート
サーバーにアクセスするためのユーザー名とパスワードを格納する資格情報 ID
SSL 構成
このクラスを拡張し、新しい dc.Demo.IMAPInboundAdapter という IMAP
インバウンドアダプタが作成されました。

この新しいアダプタを使用するために、Mailbox
本番パラメーターにどのメールボックスを使用するのかを設定します。 デフォルト値は INBOX です。

実装は単純なもので、MailServer
プロパティをオーバーライドして、そのタイプを dc.Demo.POP3ToIMAPAdapter IMAP
クライアントに設定するだけです。 基本のアダプタクラスは POP3
コマンド用に設計されているため、このアダプタは POP3 フローを IMAP フローにマッピングします。

したがって、この POP3 から IMAP
へのアダプタを使用すると、元のすべてのインバウンドアダプタロジックを実行しながら、POP3
コマンドの代わりに IMAP コマンドを使用することができます。

dc.Demo.POP3ToIMAPAdapter クラスでは、サーバー通信のプロキシとして、タイプ dc.Demo.IMAP の IMAP
クライアント IMAPClient を使用しますが、 dc.Demo.POP3ToIMAPAdapter は %Net.POP3
を拡張しているため、%Net.FetchMailProtocol のすべての抽象メソッドをオーバーライドする必要があります。

また、%Net.POP3 クライアントが直接実装していた ConnectPort と FetchMessageHeaders
という新しいメソッドを実装する必要がありました。
同様に、ConnectedGet と SSLConfigurationSet メソッドを作成して、%New.POP3
が直接実装していたプロパティを設定して取得しました。

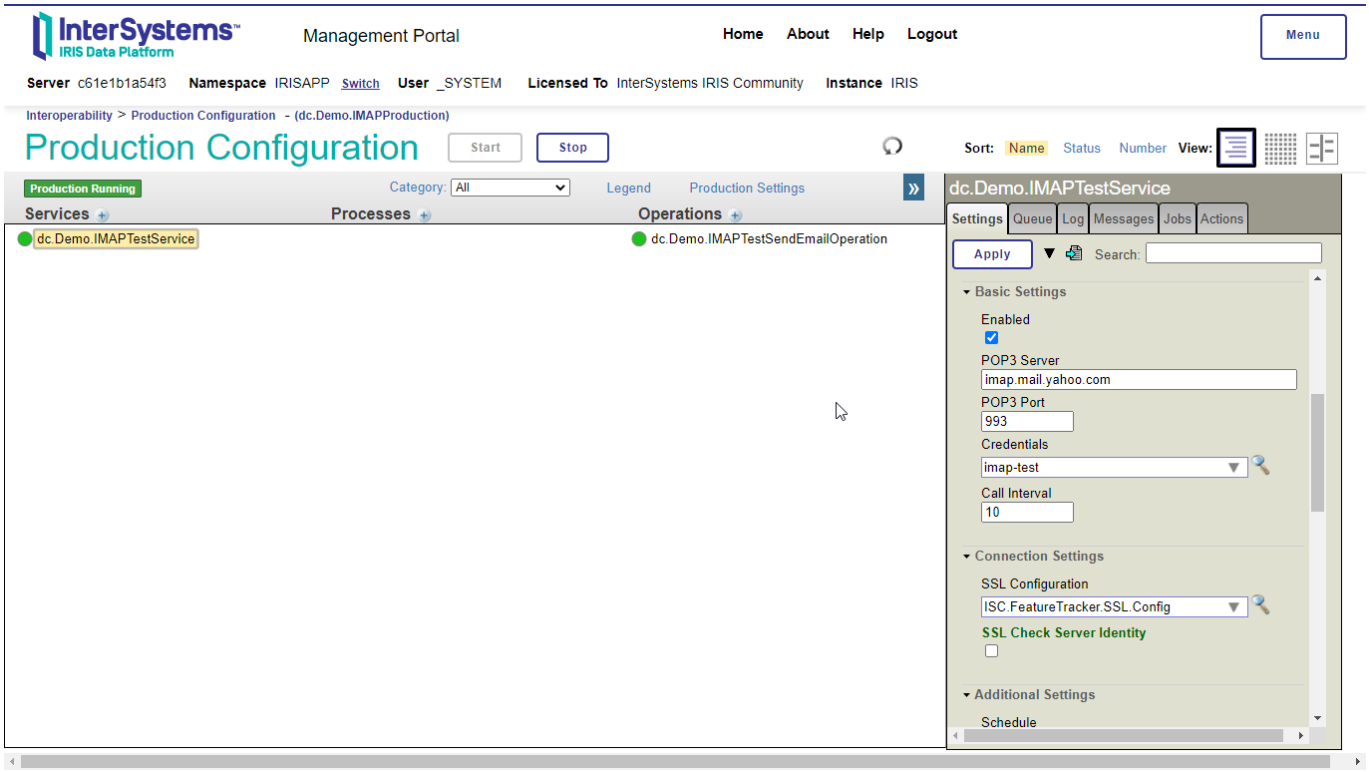
単純なプロダクションのセットアップ

すべてのクラスが連携するように、単純なプロダクションをセットアップします。 IRIS
プロダクションに関する詳細については、プロダクションの作成 をご覧ください。

このプロダクションには、IMAP
インバウンドアダプタを使って新着メッセージをチ
ェックするビジネスサービスとビジネスオペレーションが含まれています。
このコードは、Demo.Loan.FindRateProduction の相互運用性サンプルを変更したものです。

要するに、このプロダクションでは以下のことを行います。

GetMessageUIDArray
メソッドを使用して、構成済みのメールボックスに存在するすべてのメッセージを取得する
メッセージをループし、Fetch メソッドでフェッチされた出力をトレースする
各メッセージの件名が、「[IMAP test] で開始」という基準に一致しているかどうかをチェックする
メッセージの件名が基準に一致する場合は送信者に返信し、そうでない場合はメッセージを無視する
それらのメッセージを削除して、もう一度解析されないようにする



この例では、Yahoo メールの IMAP サーバーである imap.mail.yahoo.com（ポート 993）を構成します。
また、「ISC FeatureTacker.SSL.Config」というデフォルトの IRIS SSL 構成を使用します。

次に、ユーザー名とパスワードを含む、imap-test という資格情報を以下のように構成します。

以下の画像が示すように、プロダクションが開始され、IMAP サーバーに新着メッセージを照会し続けます。 新着メッセージがある場合、インバウンドアダプタは、ヘッダーや件名といった情報を取得し、その情報に基づいてプロダクションでさらにアクションを実行できるようにします。

この例では、プロダクションはメッセージの件名が「[IMAP
test]」で開始しているかどうかをチェックし、送信者にメッセージを送り返します。

メッセージがこの基準に一致していない場合は、そのまま無視されます。

まとめ

この記事では、IMAP クライアントの実装について説明しました。 最初に、IMAP
とその主要コマンドに関するいくつかの重要な背景を確認しました。 次に、クライアントと IRIS
プラットフォームからクライアントに接続する方法など、実装を詳しく説明しました。 また、IMAP
を使用するためのデフォルトの相互運用性アダプタの拡張機能と、単純なプロダクションの例も示しました。

IMAP とその設定についてさらに理解し、IRIS
への接続方法を学習したため、アプリケーションにメール機能をセットアップできるようになったことでしょう。
ここで言及した IMAP 関連トピックの詳細については、以下のリソースをご覧ください。

リソース

Atmail の「IMAP 101: Manual IMAP Sessions」
Fastmail の「Why is IMAP better than POP?」
IETF の「Internet Message Access Protocol」
IETF の「Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies」
InterSystems の「I/O Devices and Commands」
InterSystems の「Using the Email Inbound Adapter」
Nylas の「Everything you need to know about IMAP」
#InterSystems IRIS

ソースURL:https://jp.community.intersystems.com/post/intersystems-iris-%E3%81%A7-imap-%E3%82%AF%E3%83%A9%E3%82%A4%E3%82%A2%E3%83%B3%E3%83%88%E3%82%92%E5%AE%9F%E8%A3%85%E3%81%99%E3%82%8B-%E3%83%91%E3%83%BC%E3%83%88-ii