記事
Toshihiko Minamoto · 2022年3月15日 13m read

InterSystems IRIS で Python を使って IMAPクライアントを実装する

これまでの記事では、メールサーバーのメールボックスからのメッセージを処理する IMAP プロトコルの基本的な使用方法を学習しました。 とても興味深いものではありましたが、他の人が作成してすぐに利用できるようにライブラリに提供されている実装を利用することも可能です。

IRIS データプラットフォームの改善の 1 つに、同じ IRIS プロセスで ObjectScript に並行して Python コードを記述できる機能があります。 この新機能は、組み込み Python と呼ばれます。 組み込み Python を使用すると、ObjectScript コードに巨大な Python エコシステムのライブラリの力を取り込むことができます。

この記事では、imaplib というライブラリを使用して IMAP クライアントを実装し、それを IRIS Email フレームワーク に統合することにします。 また、Python エコシステムの力を借りて、組み込み Python を使用して、IRIS プラットフォームでの実際の課題を解決する方法を示す実用的な例も確認します。

ここで実装されているすべてのコードは、こちらの GitHub リポジトリの python ディレクトリにあります。

Python コードは最近の IRIS バージョンでのみ機能することに注意してください。
この例で使用されているバージョンは、2021.1.0.215.3-zpm です。 組み込み Python に関する最新情報は、こちらでフォローできます。

組み込み Python の使用

組み込み Python を使用するための鍵は、%SYS.Python クラスにあります。
このクラスを使用すると、次のことを行えます。

- Python ライブラリのインポート: ##class(%SYS.Python).Import(“ package-name ”)
- ローカルシステムへの利用可能なカスタム Python モジュール（*.py ファイル）のインポート: ##class(%SYS.Python).Import(“ module-file.py ”)
- 次のような、割り当てまたはパラメーターでの Python の組み込み型の使用:
    - Python None オブジェクト: ##class(%SYS.Python).None()
    - Python True オブジェクト: ##class(%SYS.Python).True()
    - Python False オブジェクト: ##class(%SYS.Python).False()
- ObjectScript 文字列から Python Bytes オブジェクト（8 ビット文字列）への変換: ##class(%SYS.Python).Bytes(“ ObjectScript string ”)

これらのメソッドは、Python オブジェクトを作成し、ObjectScript オブジェクトを返すため、 その Python オブジェクトのプロパティとメソッドを直接 ObejctScript コードで使用することができます。

たとえば、secrets ライブラリを使ってパスワードを生成するために、この Python レシピ をどのように実装できるか見てみましょう。

```
USER>Set string = ##class(%SYS.Python).Import("string")

USER>Set secrets = ##class(%SYS.Python).Import("secrets")

USER>ZWrite secrets // let's check what this object is...
secrets=1@%SYS.Python  ; <module 'secrets' from '/usr/irissys/lib/python3.7/secrets.p
y'>  ; <OREF>
```

```
USER>ZWrite string // same for this one...
string=2@%SYS.Python  ; <module 'string' from '/usr/irissys/lib/python3.7/string.py'>
  ; <OREF>

USER>Set alphabet = string."ascii_letters" _ string.digits // here we are accessing P
ython properties from string object

USER>Set pwd = ""

USER>For i=1:1:8 { Set pwd = pwd _ secrets.choice(alphabet) }

USER>Write pwd
Qv7HuOPV
```

上記のコードでは、Python オブジェクトからいくつかのプロパティとメソッドを使用して、ObjectScript 変数を設定しています。 ObjectScript 変数は、Python オブジェクトメソッドのパラメーターとして使用します。

埋め込み Python を使用する上でのもう 1 つの重要なポイントは、魔法のメソッドと呼ばれることもある特有の属性とメソッドです。 Python データモデル のすべてはオブジェクトであるため、これらの属性とメソッドは、Python インタープリターのインターフェースを提供します。 たとえば、次のように、__getitem__ という特殊メソッドを使用し、アイテムのインデックスを使ってリストからアイテムを取得することができます。

```
USER>Set b = ##class(%SYS.Python).Import("builtins")

USER>Set list = b.list() // creates a Python list

USER>Do list.append(1)

USER>Do list.append(2)

USER>Do list.append(3)

USER>ZWrite list
list=4@%SYS.Python  ; [1, 2, 3]  ; <OREF>

USER>w list."__getitem__"(0) // in Python, indexes are 0-based
1
USER>w list."__getitem__"(2)
3
```

同様に、リストの長さについても、__len__特殊メソッドを使用して取得することができます。

```
USER>Set listLen = list."__len__"()

USER>ZWrite listLen
listLen=3
```

これらを組み合わせて、ObjectScript を使用してリストを反復処理することができます。

```
USER>For i=0:1:(listLen - 1) { Write list."__getitem__"(i), ! }
1
```

```
2
3
```

None、True、False のような定数値を使用する必要がある場合は、%SYS.Python クラスから次のメソッドを使用することができます。

```
USER>Set none = ##class(%SYS.Python).None()

USER>Set true = ##class(%SYS.Python).True()

USER>Set false = ##class(%SYS.Python).False()

USER>ZWrite none, true, false
none=5@%SYS.Python  ; None  ; <OREF>
true=6@%SYS.Python  ; True  ; <OREF>
false=7@%SYS.Python  ; False  ; <OREF>
```

同様に、ObjectScript 文字列を Python Bytes オブジェクトに変換することができます。

```
USER>Set bytes = ##class(%SYS.Python).Bytes("This is a string")

USER>ZWrite bytes
bytes=8@%SYS.Python  ; b'This is a string'  ; <OREF>
```

最後に、カスタム Python モジュールを定義して、ObjectScript コンテキストにインポートします。

組み込み Python の使用方法に関するその他の便利なリソースは、こちらをご覧ください。 たとえば、Robert Cemper の例はお勧めです。

代替 IMAP クライアントの作成

imaplib を使用して IMAP クライアントを実装するには、通常の ObjectScript を使用します。 最初から IMAP プロトコルを実装する代わりに、メソッドを impalib メソッドでオーバーライドします。

まず、dc.demo.imap.python.IMAPPy という新しいクラスを作成します。 このクラスは、2 つのプロパティを使用して Python オブジェクトへの参照を格納します。

```
Class dc.demo.imap.python.IMAPPy Extends dc.demo.imap.IMAP
{

/// Stores the imaplib object reference
Property imaplib As %SYS.Python;

/// Stores the imaplib client instance
Property client As %SYS.Python;

...
```

次に、imaplib ライブラリを ObjectScript コンテキストのクラスコンストラクタにインポートします。

```
Method %OnNew() As %Status [ Private ]
{
    Set ..imaplib = ##class(%SYS.Python).Import("imaplib")
    Return $$$OK
}
```

これで、imaplib クラスプロパティを使用して、すべての impalib
プロパティとメソッドにアクセスできるようになりました。 最初にオーバーライドするメソッドは、Connect
メソッドです。 このメソッドは imaplib の IMAP4_SSL メソッドを使用して、IMAP
サーバーへの接続を行います。 imaplib クライアントインスタンスをクライアントプロパティとして格納します。

imapilb クライアントの login メソッドは、次のようにしてログインリクエストを認証します。

```
Method Connect(pServer As %String, pUserName As %String, pPassword As %String) As %St
atus
{
    If ..Connected Return $$$ERROR($$$ConnectedError)
    Set sc = $$$OK
    Try {
        Set ..Server = pServer
        Set ..UserName = pUserName
        Set ..client = ..imaplib."IMAP4_SSL"(..Server)
        Set resp = ..client.login(..UserName, pPassword)
        Set ..Connected = 1
    }
    Catch ex {
        Set sc = ex.AsStatus()
    }
    Return sc
}
```

次にオーバーライドするメソッドは Disconnect メソッドです。 このメソッドは、imaplib クライアントから
logout メソッドを呼び出すようになります。

```
Method Disconnect() As %Status
{
    Set sc = $$$OK
    Try {
        If ..Connected {
            Set tuple = ..client.logout()
            Set ..Connected = 0
        }
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
}
```

GetMailBoxStatus メソッドは、imaplib から select
メソッドを使用してアクセスするメールボックスを指定するようにオーバーライドされています。

```
Method GetMailBoxStatus(ByRef NumberOfMessages As %Integer, ByRef NumberOfBytes As %I
```

```
nteger) As %Status
{
    Set sc = $$$OK
    Try {
        Do ..CheckConnection()
        Set resp = ..client.select(..MailboxName)
        Set ackToken = resp."__getitem__"(0)
        Set dataArray = resp."__getitem__"(1)
        Set NumberOfMessages = dataArray."__getitem__"(0)
        Set NumberOfBytes = -1
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
}
```

このメソッドはタプルを返すため、特殊な getitem_
メソッドによって、情報を取得することができることに注意してください。
また、タプルは別のタプルを格納することができるため、再帰的に getitem_
を使用できることを忘れないでください。

次にオーバーライドするメソッドは、GetSizeOfMessages です。 このメソッドは、select
メソッドを使用して現在のメールボックスを選択し、fetch メソッドを使用して MessageNumber
パラメーターに格納されたメッセージのサイズを取得するようになります。

```
Method GetSizeOfMessages(MessageNumber As %String = "", ByRef ListOfSizes As %ArrayOf
DataTypes) As %Status
{
    Set sc = $$$OK
    Try {
        Do ..CheckConnection()
        // select the mailbox
        Set resp = ..client.select(..MailboxName)
        // hack to ensure that MessageNumber is of type %String
        Set MessageNumber = MessageNumber_""
        Set resp = ..client.fetch(MessageNumber, "(RFC822.SIZE)")
        Set ackToken = resp."__getitem__"(0)
        Set dataArray = resp."__getitem__"(1)
        Set:('$ISOBJECT($Get(ListOfSizes))) ListOfSizes = ##class(%ArrayOfDataTypes).
%New()
        Set data = dataArray."__getitem__"(0)
        Set msgIdx = +$Piece(data, " ", 1)
        Set size = +$Piece(data, " ", 3)
        Do ListOfSizes.SetAt(size, msgIdx)
    }
    Catch ex {
        Set sc=ex.AsStatus()    }
    Return sc
}
```

同様に、GetMessageUIDArray メソッドも fetch
メソッドを使用するようにオーバーライドしますが、これは、UID コードを取得するために使用します。

```
Method GetMessageUIDArray(MessageNumber As %String = "", ByRef ListOfUniqueIDs As %Ar
```

```
rayOfDataTypes) As %Status
{
    Set sc = $$$OK
    Try {
        Do ..CheckConnection()
        // select the mailbox
        Set resp = ..client.select(..MailboxName)
        Set mailboxSize = resp."__getitem__"(1)."__getitem__"(0)
        If (mailboxSize > 0) {
            // hack to ensure that MessageNumber is of type %String
            Set MessageNumber = MessageNumber_""
            // then get the mailbox UIDs
            Set param = $CASE(MessageNumber, "":"1:*", :MessageNumber)
            Set resp = ..client.fetch(param, "UID")
            Set ackToken = resp."__getitem__"(0)
            Set dataArray = resp."__getitem__"(1)
            Set len = dataArray."__len__"()
        } Else {
            Set len = 0
        }

        Set:('$ISOBJECT($Get(ListOfUniqueIDs))) ListOfUniqueIDs = ##class(%ArrayOfDat
aTypes).%New(len)
        For i = 1:1:len {
            Set data = dataArray."__getitem__"(i - 1)
            Set msgIdx = +$Piece(data, " ", 1)
            Set size = +$Piece(data, " ", 3)
            Do ListOfUniqueIDs.SetAt(size, msgIdx)
        }
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
}
```

dataArray 変数のタプルを反復処理するために、 getitem_メソッドと len_
メソッドを使用しているところに注意してください。

```
…
        Set len = dataArray."__len__"()
…
        For i = 1:1:len {
            Set data = dataArray."__getitem__"(i - 1)
            Set msgIdx = +$Piece(data, " ", 1)
            Set size = +$Piece(data, " ", 3)
            Do ListOfUniqueIDs.SetAt(size, msgIdx)
        }
```

次に、Fetch メソッドをオーバーライドします。これは、メッセージ本文全体を取得するために使用します。

```
Method Fetch(MessageNumber As %Integer, ByRef Msg As %Net.MailMessage, Delete As %Boo
lean, messageStream As %BinaryStream) As %Status
{
    Set sc = $$$OK
```

```
    Try {
        Do ..CheckConnection()
        // select the mailbox
        Set resp = ..client.select(..MailboxName)
        // hack to ensure that MessageNumber is of type %String
        Set MessageNumber = MessageNumber_""
        // get the whole message
        Set resp = ..client.fetch(MessageNumber, "BODY.PEEK[]")
        Set rawMsg = ..TransversePythonArray(resp."__getitem__"(1))

       ...
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
}
```

TransversePythonArray メソッドが存在することに注意してください。 fetch メソッドで返されるメッセージ本文は複合コレクションであるため、このコレクションを再帰的に横断して単一の文字列にフラット化するために、このメソッドを作成しています。

```
ClassMethod TransversePythonArray(pArray As %SYS.Python) As %String
{
    Set acc = ""
    If ($IsObject(pArray)) {
        Set len = pArray."__len__"()
        For i = 1:1:len {
            Set item = pArray."__getitem__"(i - 1)
            If ($IsObject(item)) {
                Set acc = acc_..TransversePythonArray(item)
            } Else {
                Set acc = acc_item
            }
            Set acc = acc_$Char(13, 10)
        }
    } Else {
        Set acc = pArray_$Char(13, 10)
    }
    Return acc
}
```

また、imaplib noop メソッドを使用するように Ping メソッドをオーバーライドします。

```
Method Ping() As %Status
{
    Set sc = $$$OK
    Try {
        Do ..CheckConnection()
        Set resp = ..client.noop()
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
```

```
}
```

オーバーライドする最後のメソッドは CommitMarkedAsDeleted メソッドです。 store メソッドと expunge メソッドを使用して、メッセージに削除マークを付けてその操作をコミットするようになります。

```
Method CommitMarkedAsDeleted() As %Status [ Internal, Private ]
{
    Set sc = $$$OK
    Try {
        Do ..CheckConnection()
        // select the mailbox
        Set resp = ..client.select(..MailboxName)
        // transverse array in inverse order to keep numbers integrity,
        // that is, ensures that when the number is deleted no other
        // message can assume such number
        Set messageNumber = $Order(..MarkedAsDeleted(""), -1)
        While (messageNumber '= "") {
            // hack to ensure that messageNumber is of type %String
            Set messageNumber = messageNumber_""
            Set resp = ..client.store(messageNumber, "+FLAGS", "\Deleted")
            Set messageNumber = $Order(..MarkedAsDeleted(messageNumber), -1)
        }
        Kill ..MarkedAsDeleted

        Set resp = ..client.expunge()
    }
    Catch ex {
        Set sc=ex.AsStatus()
    }
    Return sc
}
```

まとめ

この方法は、IRIS TCP コマンドを使用して各 IMAP
コマンドを手動で実装する必要のある元の方法に比べ、はるかに簡単に実装できます。 機能性豊かな Python ライブラリエコシステムを実際の問題に使用する優れた方法の例を確認できたので、あなたの ObjectScript アプリケーションをパワーアップし始めましょう！

参考情報

- imaplib — IMAP4 プロトコルクライアント
- 動画: Embedded Python in InterSystems IRIS: Sneak Peek
- Embedded Python: Bring the Python Ecosystem to Your ObjectScript App
- Learn Python Network Programming: Python - IMAP
- Python ドキュメント: レシピとベストプラクティス
- Python ドキュメント: データモデル
- WebSocket Client with Embedded Python
- Intersystems 開発者コミュニティ: #Python

#Embedded Python #Python #相互運用性 #InterSystems IRIS

4%BD%BF%E3%81%A3%E3%81%A6-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