記事
Minoru Horita · 2020年7月24日 1m read

Open Exchange

Python Gateway パート IV：相互運用アダプタ

この連載記事では、InterSystemsデータプラットフォーム用のPython Gatewayについて説明します。
また、InterSystems IRISからPythonコードなどを実行します。 このプロジェクトは、InterSystems IRIS環境にPythonの力を与えます。

- 任意のPythonコードを実行する
- InterSystems IRISからPythonへのシームレスなデータ転送
- Python相互運用アダプタでインテリジェントな相互運用ビジネスプロセスを構築する
- InterSystems IRISからのPythonコンテキストの保存、調査、変更、復元

その他の記事

現時点での連載計画です（変更される可能性があります）。

- パート I：概要、展望、紹介
- パート II：インストールとトラブルシューティング
- パート III：基本機能
- パート IV：相互運用アダプタ <-- 現在、この記事を参照しています
- パート V：Execute関数
- パート VI：動的ゲートウェイ
- パート VII：プロキシゲートウェイ
- パート VIII：使用事例とML Toolkit

はじめに

前回はターミナルからPython
Gatewayを試しましたが、今回は相互運用性プロダクションを介してPythonゲートウェイを使ってみましょう。
この記事では、Pythonの重要な相互運用性インターフェースである isc.py.ens.Operation について説明します。
このインターフェースにより、次のことが可能になります。

- Pythonコードを実行してリクエストされた変数を返す（文字列/ストリーム）
- コンテキストを保存/復元する
- Pythonにデータを読み込む

一般的に言えば、これは isc.py.Main の相互運用ラッパーです。 相互運用アダプタである
isc.py.ens.Operationは、相互運用性プロダクションからPythonプロセスと対話する機能を提供します。
現在、5つのリクエストがサポートされています。

- isc.py.msg.ExecutionRequest を介してPythonコードを実行します。 リクエストされた変数値を含む isc.py.msg.ExecutionResponse を返します。
- isc.py.msg.StreamExecutionRequest を介してPythonコードを実行します。
  リクエストされた変数値を含む isc.py.msg.StreamExecutionResponse を返します。
  上記と同じですが、文字列ではなくストリームを受け付けて返します。
- isc.py.msg.QueryRequest を使用してSQLクエリからデータセットを設定します。 Ens.Response

を返します。
- isc.py.msg.GlobalRequest/isc.py.msg.ClassRequest/isc.py.msg.TableRequest
を使用してグローバル/クラス/テーブルからより高速にデータセットを設定します。 Ens.Response
を返します。
- isc.py.msg.SaveRequest を介してPythonコンテキストを保存します。 コンテキストIDを含む
Ens.StringResponse を返します。
-
isc.py.msg.RestoreRequest 経由でPythonコンテキストを復元します。

ただし、isc.py.ens.Operation には次の2つの設定があります。

  - Initializer - isc.py.init.Abstract を実装するクラスを選択します。
関数、モジュール、クラスなどを読み込むために使用できます。
プロセスの開始時に実行されます。
  - PythonLib - （Linuxのみ）読み込みエラーが表示される場合、これを libpython3.6m.so
か共有ライブラリのフルパスに設定します。

ビジネスプロセスの記述

次のような、ビジネスプロセスの開発を容易にする2つのユーティリティクラスがあります。
- isc.py.ens.ProcessUtilsを使用すると、変数を置換してアノテーションを取得できます。
- isc.py.util.BPEmulator を使用すると、Pythonの相互運用ビジネスプロセスを簡単にテストできます。
現在のジョブで、ビジネスプロセス（Python部分）を実行できます。

変数の置換

isc.py.ens.ProcessUtils を継承するすべてのビジネスプロセスは、GetAnnotation(name)
メソッドを使用し、アクティビティ名を指定してアクティビティのアノテーションを取得できます。 アクティビティのアノテーションには、Pythonに渡される前にObjectScript側で計算される変数を含めることができます。
以下は、変数置換の構文です。

- ${class:method:arg1:...:argN} - メソッドを実行します
- #{expr} - ObjectScriptコードを実行します

例えば、Correlation Matrix: Graph アクティビティ:
f.savefig(r'#{process.WorkDirectory}SHOWCASE${%PopulateUtils:Integer:1:100}.png') でテスト
isc.py.test.Process ビジネスプロセスをチェックします。

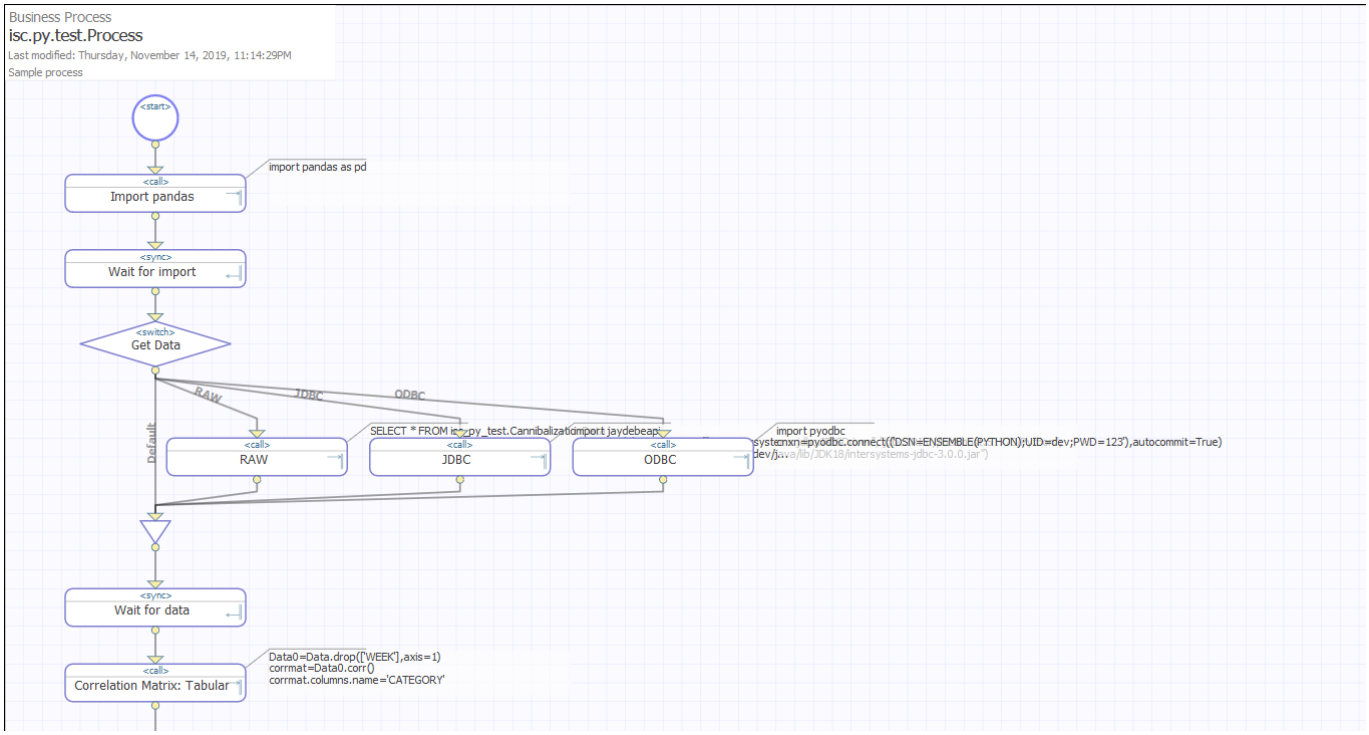
この例の場合：
- #{process.WorkDirectory} は、isc.py.test.Process クラスと現在のビジネスプロセスのインスタンスである
process オブジェクトの WorkDirectory プロパティを返します。
- ${%PopulateUtils:Integer:1:100} は 1 と 100 の引数を渡す %PopulateUtils クラスの Integer
メソッドを呼び出し、 1...100 の範囲でランダムな整数を返します。

ビジネスプロセスのテスト

テスト相互運用性プロダクションとテストビジネスプロセスは、デフォルトでPythonゲートウェイの一部として使用できます。 これらは以下の手順で使用できます。

1. OSのbashシェルで、pip install pandas matplotlib seaborn を実行します。
2. do ##class(isc.py.test.CannibalizationData).Import() を実行してテストデータを生成します。
3. isc.py.test.Production プロダクションを起動します。
4. 空の Ens.Request メッセージを isc.py.test.Process に送信します。

これらがどのように連携するか見てみましょう。 isc.py.test.Process
をBPLエディタ（またはStudio）で開いてください。



コード実行の呼び出し

以下は、Python コードを実行するのに最も重要な呼び出しです。

リクエストは isc.py.msg.ExecutionRequest であり、プロパティは次のとおりです。
- Code - 実行するPythonコードです。
- SeparateLines - 受信メッセージを実行用に複数の行に分割します。 $c(10) (\n) が行の分割に使用されます。
メッセージ全体を一度に処理することはお勧めしません。この機能は、def
や同様の複数行式の処理にのみ使用できます。 デフォルト値は0です。
- Variables - 応答メッセージで取得する変数をカンマで区切ったリストです。
- Serialization - 返したい変数のシリアライズ方法です。Str / Repr / JSON / Pickle / Dill
のオプションがあり、デフォルト値は Str です。

この例では Code
プロパティのみを設定していますので、残りのプロパティはすべてデフォルト値になっています。
私たちはこれを、実行時に変数置換を実行した後にアノテーションを返す process.GetAnnotation("Import pandas") を呼び出して設定しました。 最終的には、import pandas as pd 文字列がPythonに渡されます。
GetAnnotation は複数行のPythonスクリプトをセットアップするのに役立ちますが、制限はありません。 Code
プロパティは好きなように設定できます。

変数取得の呼び出し

他にも isc.py.msg.ExecutionRequest を使用する Correlation Matrix: Tabular
という興味深い呼び出しがあります。

これはPython側でCorrelation Matrixを計算して corrmat
を取得し、次のようにリクエストにプロパティを設定してInterSystems IRISにJSON形式で返します。
- Variables: "corrmat"
- Serialization: "JSON"

結果は次のように視覚的なトレースで確認できます。

また、これが今後ビジネスプロセスで必要な場合は、callresponse.Variables.GetAt("corrmat")
を使って保存できます。

データ転送の呼び出し

次に、InterSystems IRISからPythonへのデータ転送について説明します。すべてのデータ転送リクエストは、以下の共通プロパティを提供する isc.py.msg.DataRequest を拡張します。
- Variable - 設定するPython変数です。
- Type - 変数の型。現在は、dataframe（pandasのデータフレーム）と list がサポートされています。
- Namespace - データを取得するネームスペースです。 'isc.py'
パッケージをこのネームスペースで使用できる必要があります。

その上に以下の4つの具象クラスがあります。
- isc.py.msg.QueryRequest - SQL結果セットを転送するために Query プロパティを設定します。
- isc.py.msg.ClassRequest - クラスデータを転送するために Class プロパティを設定します。
- isc.py.msg.TableRequest - テーブル全体を転送するために Table プロパティを設定します。
- isc.py.msg.GlobalRequest - グローバルを転送するために Global プロパティを設定します。

このテストプロセスでは、RAW 呼び出しをチェックして isc.py.msg.QueryRequest の動作を確認します。

Pythonコンテキスト呼び出しの保存/復元

最後に、PythonコンテキストをInterSystems IRISに永続化します。そのためには、次を指定して
isc.py.msg.SaveRequest を送信してください。
- Mask - Mask を満たす変数のみが保存されます。 ワイルドカード * および ? を使用できます
（例："Data*,Figure?"）。 デフォルト値は * です。
- MaxLength - 保存される変数の最大長です。
変数をシリアライズした結果がこの値よりも長い場合、結果は無視されます。
すべてを取得するには、0に設定してください。 デフォルト値は $$$MaxStringLength です。
- Name - コンテキスト名です（任意）。
- Description - 拡張コンテキスト情報です（任意）。

例えば、テストプロセスでは Save Context の呼び出しをチェックします。 保存されたコンテキストの Id を含む Ens.StringResponse を返します。

対応する isc.py.msg.RestoreRequest はInterSystems IRISからPythonにコンテキストを読み込みます。
- ContextId - 復元するコンテキストの識別子。
- Clear - 復元前にコンテキストをクリアします。

要約

Python Gatewayを使用すると、InterSytems IRISとPythonをシームレスに統合できます。
これを使用し、Pythonの機能を相互運用性プロダクションに追加してください。

リンク

- Python Gateway
- Python Gatewayのサンプル
- Python 3.6.7（64ビット版）のインストール
- Pythonのドキュメント/チュートリアル

イラスト付きガイド

ML Toolkitユーザーグループには、イラスト付きのガイドもあります。 ML Toolkitユーザーグループは、InterSystems社のGitHub組織の一部として設定されている非公開GitHubリポジトリです。 このリポジトリは、Python Gatewayを含むML
Toolkitコンポーネントをインストール、学習、またはすでに使用している外部ユーザーを対象としています。 ML Toolkitユーザーグループに参加するには、以下の内容を含む簡単なメールを MLToolkit@intersystems.com 宛に送信してください（グループメンバーが議論中にあなたを認識して特定するために必要です）。

- GitHubのユーザー名
- 氏名（英文字表記の名前、姓の順）
- 組織（勤務先、通学先、または在宅勤務）
- 役職（組織での実際の役職、「学生」、または「無所属」）
- 国（本拠地としている国）

#ビジネスオペレーション #ビジネスプロセス（BPL） #相互運用性 #InterSystems IRIS
InterSystems Open Exchangeで関連アプリケーションを確認してください

ソースURL:https://jp.community.intersystems.com/post/python-gateway-%E3%83%91%E3%83%BC%E3%83%88-iv%EF%BC%9A%E7%9B%B8%E4%BA%92%E9%81%8B%E7%94%A8%E3%82%A2%E3%83%80%E3%83%97%E3%82%BF