記事
Toshihiko Minamoto · 2022年6月24日 9m read

Open Exchange

Intersystems IRIS と組み込み Python の相互運用性

1. interoperability-embedded-python

この概念実証では、**embedded Python** で **IRIS 相互運用フレームワーク**をどのように使用できるかについて示すことを目的としています。

1.1. 目次

- 1. interoperability-embedded-python
  - 1.1. 目次
  - 1.2. 例
  - 1.3. コンポーネントの登録
- 2. デモ
- 3. 前提条件
- 4. Docker を使用したインストール
- 5. Docker を使用しないインストール
- 6. サンプルの実行方法
- 7. リポジトリの内容
  - 7.1. Dockerfile
  - 7.2. .vscode/settings.json
  - 7.3. .vscode/launch.json
  - 7.4. .vscode/extensions.json
  - 7.5. src フォルダ
- 8. 新しいコンポーネントの追加方法
  - 8.1. InboundAdapter
  - 8.2. OutboundAdapter
  - 8.3. BusinessService
  - 8.4. BusinessProcess
  - 8.5. BusinessOperation
  - 8.6. コンポーネントの登録
  - 8.7. Grongier.PEX の直接利用
- 9. 今後の取り組み
- 10. クレジット

1.2. 例

```
import grongier.pex
import iris
import MyResponse

class MyBusinessOperation(grongier.pex.BusinessOperation):

    def OnInit(self):
        print("[Python] ...MyBusinessOperation:OnInit() is called")
        self.LOGINFO("Operation OnInit")
        return
```

```
    def OnTeardown(self):
        print("[Python] ...MyBusinessOperation:OnTeardown() is called")
        return

    def OnMessage(self, messageInput):
        if hasattr(messageInput,"_IsA"):
            if messageInput._IsA("Ens.StringRequest"):
                self.LOGINFO(f"[Python] ...This iris class is a Ens.StringRequest wit
h this message {messageInput.StringValue}")
        self.LOGINFO("Operation OnMessage")
        response = MyResponse.MyResponse("...MyBusinessOperation:OnMessage() echos")
        return response
```

1.3. コンポーネントの登録

ObjectScript は不要です。

これは、Grongier.PEX.Utils.RegisterComponent() メソッドを使用できるためです。

Embedded Python シェルから始めます。

```
/usr/irissys/bin/irispython
```

次に、このクラスメソッドを使用して、新しい py
ファイルを相互運用のためのコンポーネントリストに追加します。

```
iris.cls("Grongier.PEX.Utils").RegisterComponent(<ModuleName>,<ClassName>,<PathToPyFi
le>,<OverWrite>,<NameOfTheComponent>)
```

例:

```
iris.cls("Grongier.PEX.Utils").RegisterComponent("MyCombinedBusinessOperation","MyCom
binedBusinessOperation","/irisdev/app/src/python/demo/",1,"PEX.MyCombinedBusinessOper
ation")
```

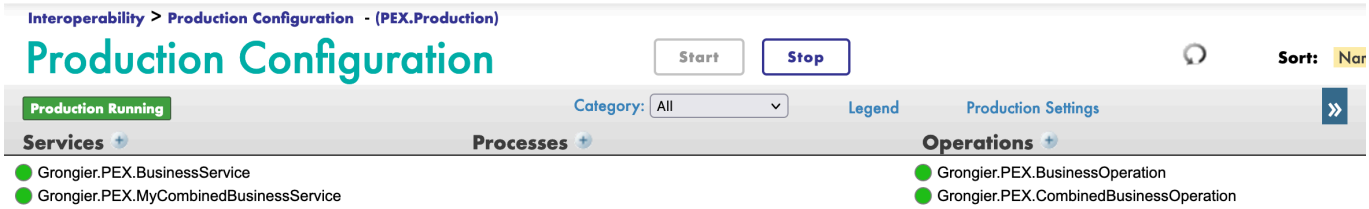
これはハックであり、本番用ではありません。

2. デモ

本番には純粋な Python によるコンポーネントが 4 つあります。

- 2 つのビジネスサービス:
    - Grongier.PEX.MyCombinedBusinessService:
      同期メッセージがビジネスオペレーションに連続して送信されます。
        - これらのメッセージは、JSON で作成されて Grongier.PEX.Message に格納される Python オブジェクトです。
        - Python コード: src/python/demo/MyCombinedBusinessService.py
    - Grongier.PEX.MyBusinessService:
      基本的に何も行いません。メッセージログを書き込む生のビジネスサービスです。

        - Python コード: src/python/demo/MyBusinessService.py
- 2 つのビジネスオペレーション:
    - Grongier.PEX.BusinessOperation: Grongier.PEX.MyCombinedBusinessService からメッセージを受け取ります。
        - Python コード: src/python/demo/MyBusinessOperation.py
    - Grongier.PEX.CombinedBusinessOperation: Ens.StringRequest メッセージを受け取り、Ens.StringResponse で応答します。
        - Python コード: src/python/demo/MyCombinedBusinessOperation.py



Python ネイティブメッセージの新しい JSON トレース:

3. 前提条件

git と Docker desktop がインストール済みであることを確認してください。

4. Docker を使用したインストール

リポジトリを任意のローカルディレクトリに Clone/git pull します。

```
git clone https://github.com/grongierisc/interpeorability-embedded-python
```

このディレクトリでターミナルを開き、以下を実行します。

```
docker-compose build
```

プロジェクトで IRIS コンテナを実行します。

```
docker-compose up -d
```

5. Docker を使用しないインストール

ローカルの IRIS インスタンスに *grongierpex-1.0.0-py3-none-any.whl* をインストールします。

```
/usr/irissys/bin/irispython -m pip install grongier_pex-1.0.0-py3-none-any.whl
```

次に、ObjectScript クラスを読み込みます。

```
do $System.OBJ.LoadDir("/opt/irisapp/src","cubk","*.cls",1)
```

6. サンプルの実行方法

Production を開いて起動します。 コードサンプルの実行が開始します。

7. リポジトリの内容

7.1. Dockerfile

便宜上、コンテナに Python の依存関係（pip、venv）と sudo をインストールする dockerfile です。
インストールすると、dev ディレクトリを作成して、この git リポジトリをそれにコピーします。

これは、IRIS を起動して Titanics cvs ファイルをインポートし、**Python Shell** 用の **%SystemCallIn** を有効にします。 関連する docker-compose.yml を使用すると、ポート番号やキーのマッピング場所、ホストフォルダなどの追加パラメーターを簡単にセットアップできます。

この dockerfile は、Python モジュールの要件をインストールして終了します。

最後に実行されるのは、Jupyter ノートブックとそのカーネルのインストールです。

.env/ ファイルを使用して、docker-compose で使用される dockerfile を調整してください。

7.2. .vscode/settings.json

VSCode ObjectScript プラグインを使って、VSCode
で直ちにコーディングできるようにするための設定ファイルです。

7.3. .vscode/launch.json

VSCode ObjectScript でデバッグする場合の構成ファイルです。

この記事に使用されるすべてのファイルについてお読みください。

7.4. .vscode/extensions.json

コンテナで VSCode を使って実行する場合に、拡張機能を追加する推奨ファイルです。

詳細情報はこちらをご覧ください。

これは、組み込み Python を使って作業する場合に非常に役立ちます。

7.5. src フォルダ

```
src
??? Grongier
?   ??? PEX // Python ????????? ObjectScript ???
?       ??? BusinessOperation.cls
?       ??? BusinessProcess.cls
?       ??? BusinessService.cls
?       ??? Common.cls
?       ??? Director.cls
?       ??? InboundAdapter.cls
?       ??? Message.cls
```

```
?       ??? OutboundAdapter.cls
?       ??? Python.cls
?       ??? Test.cls
?       ??? Utils.cls
??? PEX // ??????????
?   ??? MyBusinessOperationWithAdapter.cls
?   ??? MyBusinessOperationWithIrisAdapter.cls
?   ??? MyBusinessOperationWithPythonAdapter.cls
?   ??? MyBusinessService.cls
?   ??? MyOutboundAdapter.cls
?   ??? Production.cls
??? python
    ??? demo // ?????????????? Python ???
    ?   ??? MyBusinessOperation.py
    ?   ??? MyBusinessOperationWithAdapter.py
    ?   ??? MyBusinessOperationWithIrisAdapter.py
    ?   ??? MyBusinessProcess.py
    ?   ??? MyBusinessService.py
    ?   ??? MyCombinedBusinessOperation.py
    ?   ??? MyCombinedBusinessProcess.py
    ?   ??? MyCombinedBusinessService.py
    ?   ??? MyInboundAdapter.py
    ?   ??? MyLoggingOperation.py
    ?   ??? MyNonPollingStarter.py
    ?   ??? MyOutboundAdapter.py
    ?   ??? MyRequest.py
    ?   ??? MyResponse.py
    ?   ??? MySyncBusinessProcess.py
    ?   ??? SimpleObject.py
    ??? dist // Python ????????????????????????? Wheel
    ?   ??? grongier_pex-1.0.0-py3-none-any.whl
    ??? grongier
    ?   ??? pex // ?????????????????????????????
    ?       ??? _BusinessHost.py
    ?       ??? _BusinessOperation.py
    ?       ??? _BusinessProcess.py
    ?       ??? _BusinessService.py
    ?       ??? _Common.py
    ?       ??? _Director.py
    ?       ??? _InboundAdapter.py
    ?       ??? _Message.py
    ?       ??? _OutboundAdapter.py
    ?       ??? __init__.py
    ??? setup.py // Wheel ??????????????
```

8. 新しいコンポーネントの追加方法

8.1. InboundAdapter

Python で InboundAdapter を実装するには、以下を実行します。

Python で grongier.pex.InboundAdapter からサブクラス化します。 OnTask() メソッドをオーバーライドします。

8.2. OutboundAdapter

Python で OutboundAdapter を実装するには、以下を実行します。

Python で grongier.pex.OutboundAdapter からサブクラス化します。 必要なアクションメソッドを実装します。

8.3. BusinessService

Python で BusinessService を実装するには、以下を実行します。

Python で grongier.pex.BusinessService からサブクラス化します。 OnProcessInput()
メソッドをオーバーライドします。

8.4. BusinessProcess

Python で BusinessProcess を実装するには、以下を実行します。

Python で grongier.pex.BusinessProcess からサブクラス化します。 OnRequest()、OnResponse()、および
OnComplete() メソッドをオーバーライドします。

8.5. BusinessOperation

Python で BusinessOperation を実装するには、以下を実行します。

Python で grongier.pex.BusinessOperation からサブクラス化します。 OnMessage()
メソッドをオーバーライドします。

8.6. コンポーネントの登録

組み込み Python シェルを起動します。

```
/usr/irissys/bin/irispython
```

次に、このクラスメソッドを使用して、新しい py
ファイルを相互運用のためのコンポーネントリストに追加します。

```
iris.cls("Grongier.PEX.Utils").RegisterComponent(<ModuleName>,<ClassName>,<PathToPyFi
le>,<OverWrite>,<NameOfTheComponent>)
```

例:

```
iris.cls("Grongier.PEX.Utils").RegisterComponent("MyCombinedBusinessOperation","MyCom
binedBusinessOperation","/irisdev/app/src/python/demo/",1,"PEX.MyCombinedBusinessOper
ation")
```

8.7. Grongier.PEX の直接利用

RegisterComponent ユーティリティを使用しない場合は、 Grongier.PEX.Business*
コンポーネントを追加して、そのプロパティを構成できます。

- %module:
  - Python コードのモジュール名
- %classname:
  - コンポーネントのクラス名
- %classpaths:
  - コンポーネントのあるパス
  - これは、PYTHONPATH のほかに必要な 1 つまたは複数のクラスパスである場合があります（複数の場合は '|'（パイプ）区切り）。

例:

9. 今後の取り組み

- Service と Operation のみがテスト済みです。
- Adapter は作業中です。

10. クレジット

このコードの大部分は、Mo Cheng と Summer Gerry による Python 用 PEX を使用しています。

登録部分は、IRIS 2021.3 の未公開機能を使用しています。

#Embedded Python #Python #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/intersystems-iris-%E3%81%A8%E7%B5%84%E3%81%BF%E8%BE%BC%E3%81%BF-python-%E3%81%AE%E7%9B%B8%E4%BA%92%E9%81%8B%E7%94%A8%E6%80%A7