

文章

[Michael Lei](#) · 十二月 16, 2021 阅读大约需 10 分钟

[Open Exchange](#)

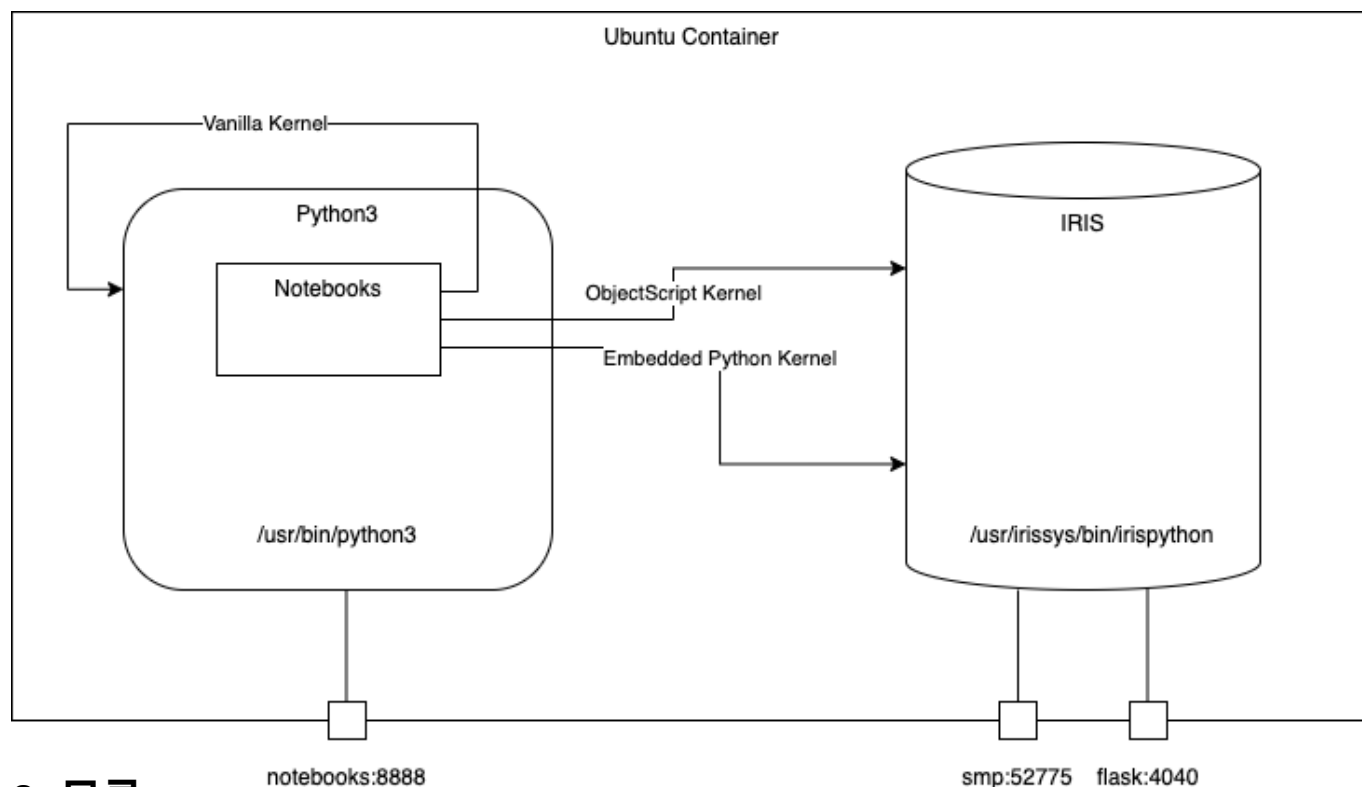
翻译文章--InterSystems IRIS 2021.2+ Python 代码样例 (Embedded嵌入式Python, Native 原生APIs 和 Notebooks)

Iris-python-template

包含各种Python代码的项目模版，可用于InterSystems IRIS 社区容器版Community Edition with container。

特性：

- Notebooks 记事本
 - Embedded Python 内核
 - ObjectScript 内核
 - Vanilla Python 内核
- Embedded嵌入式 Python
 - Code example代码样例
 - Flask demo
- IRIS Python Native 原生APIs
 - Code example



2. 目录

- [1. iris-python-template模版](#)
- [2. 目录](#)
- [3. 安装](#)
 - [3.1. Docker](#)

- [4. 开始编程](#)
 - [4.1. 预设条件Prerequisites](#)
 - [4.1.1. 使用 ObjectScript编程](#)
 - [4.1.2. 使用嵌入式Embedded Python](#)
 - [4.1.3. 使用记事本Notebooks编程](#)
- [5. Repository的内容](#)
 - [5.1. Dockerfile](#)
 - [5.2. .vscode/settings.json](#)
 - [5.3. .vscode/launch.json](#)
 - [5.4. .vscode/extensions.json](#)
 - [5.5. src folder](#)
 - [5.5.1. src/ObjectScript](#)
 - [5.5.1.1. src/ObjectScript/Embedded/Python.cls](#)
 - [5.5.1.2. src/ObjectScript/Gateway/Python.cls](#)
 - [5.5.2. src/Python](#)
 - [5.5.2.1. src/Python/embedded/demo.cls](#)
 - [5.5.2.2. src/Python/native/demo.cls](#)
 - [5.5.2.3. src/Python/flask](#)
 - [5.5.2.3.1. How it works](#)
 - [5.5.2.3.2. Launching the flask server](#)
 - [5.5.3. src/Notebooks](#)
 - [5.5.3.1. src/Notebooks/HelloWorldEmbedded.ipynb](#)
 - [5.5.3.2. src/Notebooks/IrisNative.ipynb](#)
 - [5.5.3.3. src/Notebooks/ObjectScript.ipynb](#)

3. 安装

3.1. Docker

这个 repo已经docker化，你可以clone/git 把repo拉到任何本地目录下

```
git clone https://github.com/grongierisc/iris-python-template.git
```

打开该目录下的终端并执行:

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

并为记事本Notebooks打开 <http://localhost:8888/tree>

或, 在VSCode打开克隆的目录, 启动docker-compose 并通过VSCode 菜单打开URL:

4. 如何开始编程

4.1. 前提

确保安装好 [git](#) and [Docker desktop](#)

这个repository 已经可以开始在带ObjectiveScript 插件的VSCode中编写代码.
安装 [VSCode](#), [Docker](#) and [ObjectScript](#) 插件并打开文件夹.

4.1.1. 用ObjectScript编程

打开 /src/ObjectScript/Embedded/Python.cls 类并开始尝试变更 - 它会在运行IRIS docker container容器中被编译.

4.1.2. 用嵌入式Embedded Python编程

最简单的方法是在容器中运行VsCode.

要连上一个 Docker container, 要么从命令板中选择 Remote-Containers: Attach to Running Container... (kbstyle(F1)) 或者使用在活动Activity条中选择 Remote Explorer 并从 Containers 视图选择 Attach to Container动作在你希望连接的容器上.

然后配置你的python 解析器 /usr/irissys/bin/irispython

4.1.3. 用Notebooks编程

打开这个网址 : <http://localhost:8888/tree>

你可以用三个不同的内核访问三个不同的notebooks

- Embedded嵌入式 Python 内核 kernel
- ObjectScript 内核kernel
- Vanilla python3 内核kernel

5. Repository的内容

5.1. Dockerfile

一个安装了某些python 依赖 (pip, venv) 和 容器Sudo的dockerfile 以方便使用.

接下来创建开发目录并拷贝在这个 git repository里

启动IRIS 并倒入Titanics csv 文件, 然后激活 %ServiceCallIn for Python Shell.

使用相关的docker-compose.yml 文件来轻松设置另外的参数如 port number 以及你在哪里配置 keys 和host 文件夹.

dockerfile 以安装python模块所需要的东西为终止.

最后一部分是关于安装jupyter notebook记事本和它的内核.

使用 .env/ 文件 来调整在docker-compose里面使用到的dockerfile.

5.2. .vscode/settings.json

配置文件可以马上用 [VSCode ObjectScript plugin](#)开始编程

5.3. .vscode/launch.json

配置文件如果你想用VSCode ObjectScript debug.

[Read about all the files in this article](#)

5.4. .vscode/extensions.json

如果你想在容器中运行VSCode添加拓展建议的文档.

[更多信息请查看这里](#)

这在使用embedded python非常有用.

5.5. src folder文件夹

这个文件夹被分成两部分, one 用来保存 ObjectScript 样例, 另一个保存Python 代码.

5.5.1. src/ObjectScript

显示如何在IRIS中使用python的不同部分代码.

5.5.1.1. src/ObjectScript/Embedded/Python.cls

所有的注释 (都是用法文写的因为是法国人编的代码。。。).

```
/// Embedded python example
Class ObjectScript.Embedded.Python Extends %SwizzleObject
{

    /// HelloWorld with a parameter
    ClassMethod HelloWorld(name As %String = "toto") As %Boolean [ Language = python ]
    {
        print("Hello",name)
        return True
    }

    /// Description
    Method compare(modèle, chaîne) As %Status [ Language = python ]
    {
        import re

        # compare la chaîne [chaîne] au modèle [modèle]
        # affichage résultats
        print(f"\nRésultats({chaîne},{modèle})")
        match = re.match(modèle, chaîne)
        if match:
            print(match.groups())
        else:
            print(f"La chaîne [{chaîne}] ne correspond pas au modèle [{modèle}]")
    }

    /// Description
    Method compareObjectScript(modèle, chaîne) As %Status
    {
        w !,"Résultats("_chaîne_", "_modèle_)",!
        set matcher=##class(%Regex.Matcher).%New(modèle)
        set matcher.Text=chaîne
        if matcher.Locate() {
            write matcher.GroupGet(1)
        }
        else {
            w "La chaîne ["_chaîne_] ne correspond pas au modèle ["_modèle_]"
        }
    }
}
```

```
/// Description
Method DemoPyhtonToPython() As %Status [ Language = python ]
{
    # expression régulières en python
    # récupérer les différents champs d'une chaîne
    # le modèle : une suite de chiffres entourée de caractères quelconques
    # on ne veut récupérer que la suite de chiffres
    modèle = r"^.*?(\d+).*$"

    # on confronte la chaîne au modèle
    self.compare(modèle, "xyz1234abcd")
    self.compare(modèle, "12 34")
    self.compare(modèle, "abcd")
}
```

```
Method DemoPyhtonToObjectScript() As %Status [ Language = python ]
{
    # expression régulières en python
    # récupérer les différents champs d'une chaîne
    # le modèle : une suite de chiffres entourée de caractères quelconques
    # on ne veut récupérer que la suite de chiffres
    modèle = r"^.*?(\d+).*$"

    # on confronte la chaîne au modèle
    self.compareObjectScript(modèle, "xyz1234abcd")
    self.compareObjectScript(modèle, "12 34")
    self.compareObjectScript(modèle, "abcd")
}
```

```
/// Description
Method DemoObjectScriptToPython() As %Status
{
    // le modèle - une date au format jj/mm/aa
    set modèle = "^\\s*(\\d\\d)\\/(\\d\\d)\\/(\\d\\d)\\s*$"
    do ..compare(modèle, "10/05/97")
    do ..compare(modèle, " 04/04/01 ")
    do ..compare(modèle, "5/1/01")
}
}
```

- HelloWorld
 - *用python简单地打个招呼吧
 - *在标签Tag熵使用ObjectScript wrapper打包器 [Language = python]
- 对比
 - 一个用来对比带有regex的字符串的python 函数, 如果匹配就打印, 否则如果找不到匹配就不打印
- compareObjectScript
 - ObjectScript 中跟Python 一样的函数
- DemoPyhtonToPython
 - 显示如何在ObjectScript中打包的python代码中使用python函数

```
set demo = ##class(ObjectScript.Embbded.Python).%New()
```

```
zw demo.DemoPyhtonToPython()
```

- DemoPyhtonToObjectScript

- 显示如何调用ObjectScript 函数的python函数
- DemoObjectScriptToPython
 - 显示如何调用python函数的ObjectScript函数（好像绕口令，哈哈）

5.5.1.2. src/ObjectScript/Gateway/Python.cls

显示如何用gateway功能调用外部python 代码的ObjectiveScript 类.

在这个栗子中python 代码并不在同一IRIS进程中“被执行”.

```
/// Description
Class Gateway.Python
{

/// Demo of a python gateway to execute python code outside of an iris process.
ClassMethod Demo() As %Status
{
    Set sc = $$$OK

    set pyGate = $system.external.getPythonGateway()

    d pyGate.addToPath("/irisdev/app/src/Python/gateway/Address.py")

    set objectBase = ##class(%Net.Remote.Object).%New(pyGate,"Address")

    set street = objectBase.street

    zw street

    Return sc
}
}
```

5.5.2. src/Python

显示如何在IRIS中使用嵌入式embedded python的不同部分的python 代码.

5.5.2.1. src/Python/embedded/demo.cls

所有的注释（都是用法文写的因为是法国人编的代码。。。）

```
import iris

person = iris.cls('Titanic.Table.Passenger')._OpenId(1)

print(person.__dict__)
```

首先倒入iris 模块来启用 嵌入式embedded python 能力.
从IRIS模块中打开一个带cls功能的持久化类 class.
请注意所有 % 功能被替换为 _

你需要使用shell来运行这个例子：

```
/usr/irissys/bin/irispython /opt/irisapp/src/Python/embedded/demo.py
```

5.5.2.2. src/Python/native/demo.cls

显示如何在python代码中使用native api .

```
import irisnative

# create database connection and IRIS instance
connection = irisnative.createConnection("localhost", 1972, "USER", "superuser", "SYS",
sharedmemory = False)
myIris = irisnative.createIris(connection)

# classMethod
passenger = myIris.classMethodObject("Titanic.Table.Passenger", "%OpenId", 1)
print(passenger.get("name"))

# global
myIris.set("hello", "myGlobal")
print(myIris.get("myGlobal"))
```

为了倒入 irisnative, 你需要在python环境中安装 native api wheels.

```
pip3 install /usr/irissys/dev/python/intersystems_irispython-3.2.0-py3-none-any.whl
```

然后你可以执行python代码

```
/usr/bin/python3 /opt/irisapp/src/Python/native/demo.py
```

请注意在这个例子中有一个连接是练到IRIS数据库的, 这意味着, **这个代码是在另一个不同的IRIS 线程中被执行.**

5.5.2.3. src/Python/flask

一个完整的结合嵌入式embedded python和微框架flask的demo.
你可以测试一下 :

```
GET http://localhost:4040/api/passengers?currPage=1&pageSize=1
```

5.5.2.3.1. 它是如何工作的

为了使用嵌入式embedded Python, 我们使用 irispython 作为python 解析 并do:

```
import iris
```

就在文件的最开头.

我们将能够运行如下的方法methods:

正如你们看到的 , 为了 GET passenger 的 ID, 我们需要执行查询并使用它的结果集.

我们也可以直接用IRIS的对象:

在这里, 我们使用SQL 查询来获得所有表里的 ID, 并从Titanic.Table.Passenger 类带有 %OpenId() 方法method的表里获取每个passenger (请注意由于 % 在 Python是非法字符, 我们用下划线 _来代替).

感谢Flask, 我们已经这样完成了所有的方法和路线.

5.5.2.3.2. 启动 flask server

为了启动服务器, 我们用 gunicorn 和 irispthon.

在 docker-compose 文件里, 我们增加了下面的行:

```
iris:
  command: -a "sh /opt/irisapp/server_start.sh"
```

这会在容器启动后(感谢 -a flag), 启动以下脚本:

```
#!/bin/bash

cd ${SRC_PATH}/src/Python/flask

${PYTHON_PATH} -m gunicorn --bind "0.0.0.0:8080" wsgi:app &

exit 1
```

Dockerfile 里的环境参数如下:

```
ENV PYTHON_PATH=/usr/irissys/bin/irispthon
ENV SRC_PATH=/opt/irisapp/
```

5.5.3. src/Notebooks

带有三种不同内核的三个不同的记事本Notebooks :

- 一个跑原生API的 Python3 内核
- 一个嵌入式Embedded Python 内核
- 一个 ObjectScript 内核

记事本Notebooks 在这里访问 <http://localhost:8888/tree>

5.5.3.1. src/Notebooks/HelloWorldEmbedded.ipynb

这个是使用IRIS 嵌入式embedded python 内核的记事本notebook .

它展示了打开和保存持久化类和如何执行sql 查询.

5.5.3.2. src/Notebooks/IrisNative.ipynb

这个notebook使用 vanilla python 内核.

它展示了如何执行IRIS原生 native apis.

5.5.3.3. src/Notebooks/ObjectScript.ipynb

这个notebook 使用 ObjectScript 内核kernel.

它展示了如何运行ObjectScript代码以及如何在ObjectScript 中使用嵌入式embedded python.

[#Python #InterSystems IRIS](#)

[在 InterSystems Open Exchange 上检查相关应用程序](#)

源

URL:

<https://cn.community.intersystems.com/post/%E7%BF%BB%E8%AF%91%E6%96%87%E7%AB%A0-intersystems-iris-20212-python-%E4%BB%A3%E7%A0%81%E6%A0%B7%E4%BE%8B-embedded%E5%B5%8C%E5%85%A5%E5%BC%8Fpython-native-%E5%8E%9F%E7%94%9Fapis-%E5%92%8C-notebooks>