Artigo
Yuri Marx Perei... · Mar. 23, 2022 18min de leitura

Introdução ao Embbeded Python - Um processador de imagens como exemplo

A partir da versão 2021.2 do InterSystems IRIS é possível desenvolver serviços de backend, de integração e procedures de bancos de dados utilizando Python. A grande vantagem desta possibilidade é a redução na curva de aprendizado e a utilização de programadores especialistas na linguagem de programação que mais cresce no mundo. O propósito deste artigo é de demonstrar que os projetos criados em InterSystems IRIS podem ser desenvolvidos com Python, ou mesmo com Python e ObjectScript (linguagem de programação proprietária da InterSystems) juntos, para atender a quaisquer requisitos e necessidades do negócio. O desafio com este artigo é processar imagens de forma geral usando Python, uma vez que o ObjectScript não conta com funcionalidades nativas para isto.

Para ilustrar na prática este casamento perfeito do Python e do InterSystems IRIS, foi criada uma aplicação de exemplo, o iris-image-editor. Esta aplicação possui as seguintes funcionalidades:

- Criação de thumbnail (miniatura) na imagem;
- Criação de marca d´agua na imagem;
- Criação de filtros na imagem, como filtro blur, dentre outros.

Esta aplicação utiliza a biblioteca da comunidade Python Pillow (https://pillow.readthedocs.io/en/stable/). Ela permite realizar desde de operações básicas (leitura, gravação, corte, ampliação, redução, etc.), até as mais avançadas em imagens (aplicação de filtros em geral).

Para obter e executar o iris-image-editor, siga os passos abaixo:

1. Clone o repositório para um diretório local à sua escolha

```
$ git clone https://github.com/yurimarx/iris-image-editor.git
```

2. Abra o terminal do docker no diretório escolhido e execute:

```
$ docker-compose build
```

3. Execute o container do IRIS:

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

4. **Para criar thumbnails (miniaturas)**: Vá no seu Postman (ou cliente REST similar) e configure a requisição como na imagem a seguir, envie e veja a resposta:

- Method: POST
- URL: http://localhost:52773/iris-image-editor/thumbnail
- Body: form-data
- Key: file (o nome do campo file deve ser file) e o type File
- Value: arquivo do seu computador

5. **Para fazer a marca d'agua**: Vá no seu Postman (ou outro cliente REST similar) e configure a requisição como na figura, envie e veja o resultado:

- Method: POST
- URL: http://localhost:52773/iris-image-editor/watermark
- Body: form-data
- Key: file (o nome do campo file deve ser file) e o tipo File
- Value: arquivo do seu computador
- Key: watermark (texto a ser escrito dentro da imagem) e o type é text
- Value: I'm a cat (ou outro valor que você queira)

6. **Para fazer filtros**: Vá no seu Postman (ou outro cliente REST) e configurar a requisição como na imagem, envie e veja a resposta:

- Method: POST
- URL: http://localhost:52773/iris-image-editor/filter
- Body: form-data
- Key: file (o nome do campo deve ser file) e o type File
- Value: arquivo do seu computador
- Key: filter (deve ser este nome filter) e o tipo text
- Value: BLUR, CONTOUR, DETAIL, EDGEENHANCE, EDGEENHANCEMORE, EMBOSS, FINDEDGES, SMOOTH, SMOOTHMORE ou SHARPEN

Como o Python foi utilizado dentro do IRIS neste exemplo:

A primeira ação a ser tomada é ter o Python instalado junto ao IRIS, no nosso exemplo, utilizamos Docker para instalar e executar o IRIS. Sendo assim, incluimos a instalação do Python e da biblioteca Pillow no arquivo Dockerfile, veja:

Trecho de código do Dockerfile responsável por instalar o Python, o PIP e o Pillow

O apt-get install da biblioteca do Linux python3-pip, realiza a instalação do Python e do gerenciador de pacotes/bibliotecas do Python (como o ZPM do ObjectScript ou o Maven do Java), o PyPI (pip3). A partir do pip3 é possível instalar qualquer biblioteca conhecida do mundo Python, inclusive o Pillow.

O comando pip3 install --target /usr/irissys/mgr/python Pillow instala o Pillow e suas dependências dentro do diretório no qual o IRIS procura bibliotecas Python para utilizar (quando no ambiente Linux).

Uma vez que temos o Python e as bibliotecas necessárias instaladas, podemos agora criar nossos métodos de classe em linguagem Python, ao invés de utilizar ObjectScript. Neste exemplo criamos três métodos, todos na classe dc.imageeditor.ImageEditorService, detalhados abaixo:

Método de Classe em Python para gerar thumbnails para imagens

O primeiro passo é declarar Language = python na declaração do método, assim o IRIS irá utilizar Python, ao invés de ObjectScript, que a linguagem padrão. A seguir, basta escrever todo o conteúdo do método em Python, como seria feito dentro de um arquivo .py.

Este método importa o Pillow com o from PIL import Image. Em seguida são definidos os diretórios de origem da imagem e de destino da imagem processada. O Image.open, obtêm a imagem a ser processada. O image.thumbnail reduz o tamanho da imagem para as dimensões 90x90. O image.save grava o processamento no diretório de destino.

Método de Classe em Python que escreve um texto como marca d'agua da imagem

O Image.open() abre a imagem de origem. O im.size obtêm as dimensões da imagem. O ImageDraw.Draw obtem uma referência chamada draw que irá permitir a operação de escrita na imagem. O ImageFont.truetype() escolhe a fonte e tamanho do texto a ser escrito. É calculada a posição x,y do texto na image e o draw.text escreve o texto na fonte escolhida. A imagem final é gravada no diretório de destino.

Método de Classe em Python para aplicar filtros avançados na imagem

O Image.open() abre a imagem de origem. O im.filter(NOME DO FILTRO) aplica o filtro e retorna uma refêrencia para a imagem processada. A imagem com o filtro aplicado é gravada no diretório de destino com imFiltered.save().

A interação entre o ObjectScript e o Python

O exemplo também demonstra como é fácil para o ObjectScript realizar uma chamada para um método escrito em Python. É da mesma forma como seria chamado qualquer outro método. Veja este trecho de exemplo:

```
ClassMethod DoThumbnail(Image As %String) As %Status
{
  Do ..ProcessThumbnail(Image)
  Quit $$$OK
}
```

O método de classe escrito em Python está na mesma classe deste método de classe em ObjectScript, então basta utilizar .. seguido do nome do método e seus argumentos de chamada. O IRIS internamente realiza as conversões de tipos na ida e na volta para os argumentos.

Se o método estiver em outra classe, basta utilizar Do ##class(pacote.NomeClasse).NomeMetodoPython(<argumentos>).

Expondo as funcionalidades como API REST

O IRIS possui extrema facilidade em expor métodos de classe como API REST, basta ter uma classe que herde de %CSP.REST. Nesta aplicação de exemplo isto foi feito a partir da classe dc.imageeditor.ImageEditorRESTApp. Veja:

Classe responsável por expor as funcionalidades como API REST

A classe extende de %CSP.REST e possui uma seção <Routes> responsável por criar as rotas (URL) HTTP para utilizar as funcionalidades. Na tag <Route> é definida a propriedade Url como o caminho HTTP utilizado para chamar o método de classe indicado na propriedade Call. O verbo HTTP é escolhido na propriedade Method.

A chamada %request.GetMimeData() é responsável por receber os valores de input da requisição. O %response.ContentType define o tipo de retorno da requisição e a chamada %Stream.FileBinary.%New() permite obter uma referência para um arquivo, basta requisitar LinkToFile(), indicando o caminho para o arquivo. Por último é executado o OutputToDevice() para escrever o conteúdo do arquivo na resposta HTTP.

Outras formas de utilizar o Python e IRIS juntos (fonte: https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls...)

A partir do terminal do IRIS, é possível acionar o ambiente de execução do Python (shell):

```
USER>do ##class(%SYS.Python).Shell()
```

A partir de um programa Python é possível chamar Classes e Métodos ObjectScript

```
# import the iris module and show the classes in this namespace
import iris
print('\nInterSystems IRIS classes in this namespace:')
status = iris.cls('%SYSTEM.OBJ').ShowClasses()
print(status)
```

Ou

```
>>> import iris
>>> status = iris.cls('User.EmbeddedPython').Test()
Fibonacci series:
0 1 1 2 3 5 8
InterSystems IRIS classes in this namespace:
User.Company
User.EmbeddedPython
User.Person
>>> print(status)
1
```

Em funções e Stored Procedures SQL:

```
CREATE FUNCTION tzconvert(dt DATETIME, tzfrom VARCHAR, tzto VARCHAR)
    RETURNS DATETIME
    LANGUAGE PYTHON
{
    from datetime import datetime
    from dateutil import parser, tz
    d = parser.parse(dt)
    if (tzfrom is not None):
        tzf = tz.gettz(tzfrom)
        d = d.replace(tzinfo = tzf)
    return d.astimezone(tz.gettz(tzto)).strftime("%Y-%m-%d %H:%M:%S")
}
```

A partir da classe utilitária %SYS.Python

```
Class Demo.PDF
```

```
{

ClassMethod CreateSamplePDF(fileloc As %String) As %Status
{
    set canvaslib = ##class(%SYS.Python).Import("reportlab.pdfgen.canvas")
    set canvas = canvaslib.Canvas(fileloc)
    do canvas.drawImage("C:\Sample\isc.png", 150, 600)
    do canvas.drawImage("C:\Sample\python.png", 150, 200)
    do canvas.setFont("Helvetica-Bold", 24)
    do canvas.drawString(25, 450, "InterSystems IRIS & Python. Perfect Together.")
    do canvas.save()
}

}
```

Usando o Python Buitin Functions

```
set builtins = ##class(%SYS.Python).Import("builtins")



USER>do builtins.print("hello world!")
hello world!
```

Saiba mais:

- Documentação oficial: https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls...
- Concurso passado sobre Python com excelentes exemplo de uso: https://openexchange.intersystems.com/contest/21
- Outros recursos:
    - Learning Path Writing Python Application with InterSystems
    - Embedded Python Documentation
    - Native API for Python Documentation
    - PEX Documentation

#Embedded Python #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/introdu%C3%A7%C3%A3o-ao-embbeded-python-um-processador-de-imagens-como-exemplo