Artículo
Muhammad Waseem · 28 jul, 2022 Lectura de 4 min

Open Exchange

Cómo añadí ObjectScript a Jupyter Notebook

Jupyter Notebook es un entorno interactivo formado por celdas que permiten ejecutar código en un gran número de lenguajes de marcado y programación diferentes.

Para hacer esto, Jupyter debe conectarse a un kernel apropiado. No había un Kernel ObjectScript, por lo que decidí crear uno.

Puedes probarlo aquí.

Este es un adelanto de los resultados:

Jupyter Kernels 101

Hay varias formas de crear un Jupyter Kernel. Decidí hacer un kernel contenedor de Python.

Tenemos que crear una subclase de ipykernel.kernelbase.Kernel e implementar el método doexecute que recibe un código para ser ejecutado en un idioma en particular.

Entonces, la idea general es obtener un fragmento de código ObjectScript, ejecutarlo de alguna manera y devolver los resultados a nuestro cuaderno.

Pero, ¿cómo hacemos eso exactamente? Vamos a tratar de analizarlo un poco más.

Envío de código ObjectScript a IRIS

Para empezar, tenemos que enviar nuestro fragmento de código a IRIS. Aquí es donde la API Nativa de IRIS para Python es necesaria.

Todo lo que tenemos que hacer es importar el paquete irisnative, luego establecer una conexión:

```
def get_iris_object():
  # Crear conexión con InterSystems IRIS
  connection = irisnative.createConnection('iris', 51773, 'IRISAPP', '_SYSTEM', 'SYS'
)

  # Crear un objeto de iris
  return irisnative.createIris(connection)
```

Después de eso, podemos usar la conexión para llamar a las clases que están almacenadas en la base de datos de IRIS.

```
def execute_code(self, code):
        class_name = "JupyterKernel.CodeExecutor"
        return self.iris.classMethodValue(class_name, "CodeResult", code)
```

¿Para qué se utilizan esta clase Codes Executor y el método CodeResult?

Vamos a verlo.

Ejecutar código ObjectScript

El propósito de esta clase es ejecutar una línea de código ObjectScript y devolver un objeto JSON con los resultados de la ejecución. Pasamos nuestro código a CodeResult en una variable vstrCommand.

Comenzamos redirigiendo IO a la rutina actual, luego ejecutamos el código pasado a través del comando XECUTE , redirigimos IO de vuelta al original y devolvemos los resultados.

```
Include %sySystem

Class JupyterKernel.CodeExecutor
{

ClassMethod CodeResult(vstrCommand As %String) As %String [ ProcedureBlock = 0 ]
{
        set tOldIORedirected = ##class(%Device).ReDirectIO()
        set tOldMnemonic = ##class(%Device).GetMnemonicRoutine()
        set tOldIO = $io
        try {
            set str=""
            set status = 1
            //Redirigir IO a la rutina actual: utiliza las etiquetas definidas a cont
inuación
            use $io::("^"_$ZNAME)

            //Habilitar redirección
            do ##class(%Device).ReDirectIO(1)

            XECUTE (vstrCommand)

        } catch ex {
            set str = ex.DisplayString()
            set status = 0
        }

        //Vuelva a la configuración original de redirección/rutina mnemotécnica
        if (tOldMnemonic '= "") {
            use tOldIO::("^"_tOldMnemonic)
        } else {
            use tOldIO
        }
        do ##class(%Device).ReDirectIO(tOldIORedirected)

        quit {"status":(status), "out":(str)}.%ToJSON()

rchr(c)
    quit
rstr(sz,to)
    quit
wchr(s)
    do output($char(s))
```

```
    quit
wff()
    do output($char(12))
    quit
wnl()
    do output($char(13,10))
    quit
wstr(s)
    do output(s)
    quit
wtab(s)
    do output($char(9))
    quit
output(s)
    set str = str _ s
    quit
}

}
```

Mostrar los resultados

Hemos ejecutado una parte del código ObjectScript, ¿y ahora qué? Bueno, tenemos que mostrar los resultados.

Si no hubo excepciones, simplemente mostramos los resultados línea por línea.

Sin embargo, si nuestro fragmento de código aprobado generó una excepción, detenemos la ejecución, mostramos el número de la línea fallida, a sí misma y la excepción generada.

Lanzamiento de la aplicación

Puedes probar este kernel por ti mismo y aquí te mostramos cómo.

Requisitos previos

Asegúrate de tener git y Docker instalado.

Clonar/extraer el repositorio en cualquier directorio local, por ejemplo, como se muestra a continuación:

```
$ git clone https://github.com/Vekkby/objectsriptkernel.git
```

Abre el terminal en este directorio y ejecuta:

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

Cómo trabajar con él

Puedes acceder al servidor del notebook desde el navegador usando

```
localhost:8888
```

Hay un notebook de muestra llamado 'hello.ipynb' en el directorio 'work'.

#API #Python #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange