Article
Eduard Lebedyuk · Jan 6, 2020 1m read

Open Exchange

Python Gateway V: Execute function

This series of articles would cover Python Gateway for InterSystems Data Platforms. Execute Python code and more from InterSystems IRIS. This project brings you the power of Python right into your InterSystems IRIS environment:

- Execute arbitrary Python code
- Seamlessly transfer data from InterSystems IRIS into Python
- Build intelligent Interoperability business processes with Python Interoperability Adapter
- Save, examine, modify and restore Python context from InterSystems IRIS

Other articles

The plan for the series so far (subject to change).

- Part I: Overview, Landscape, and Introduction
- Part II: Installation and Troubleshooting
- Part III: Basic functionality
- Part IV: Interoperability Adapter
- Part V: Execute function<-- you're here
- Part VI: Dynamic Gateway
- Part VII: Proxy Gateway
- Part VIII: Use cases and ML Toolkit

Intro

This and next article describe the functionality pertaining to what is commonly understood as (.Net/Java) Gateway. Today we'll talk about Execute Function - ability to execute Python code programmatically in a structured way. It can be useful if you want to leverage Python code from the InterSystems IRIS application.

Execute function

Executes function by name. This API consists of two methods:
- ExecuteFunction
- ExecuteFunctionArgs

The difference between them is the signature. ExecuteFunction accepts %List, %Collection.AbstractArray and Dynamic Object separated into positional and keyword arguments. ExecuteFunctionArgs accepts args... and parses them into positional and keyword arguments. After that ExecuteFunctionArgs calls ExecuteFunction.

It is the caller responsibility to escape argument values. Use isc.py.util.Converter class to escape:
- string
- boolean
- date
- time
- timestamp

ExecuteFunction

ExecuteFunction method from isc.py.Main class. Signature:
- function - name of function to invoke. Can be nested, i.e. random.randint
- variable - name of python variable to write result to.
- positionalArguments - positional arguments for Python function. Can be one of:
  + $lb(val1, val2, ..., valN)
  + %Collection.AbstractIterator object
  + JSON array
- keywordArguments - keyword arguments for Python function. Can be one of:
  + $lb($lb(name1, val1), $lb(name2, val2), ..., $lb(nameN, valN))
  + %Collection.AbstractArray object
  + flat JSON object
- serialization - how to serialize result
- result - write result into this variable

All arguments besides function are optional.

Here's an example of how it works:

```
set sc = ##class(isc.py.Main).ImportModule("random", ,.random)

set posList = $lb(1, 100)
set posCollection = ##class(%ListOfDataTypes).%New()
do posCollection.Insert(1)
do posCollection.Insert(100)
set posDynamic = [1, 100]

for positionalArguments = posList,posCollection,posDynamic {
    set sc = ##class(isc.py.Main).ExecuteFunction(random _ ".randint", positionalArgu
ments,,,.result)
    write result,!
}

set kwList = $lb($lb("a", 1), $lb("b", 100))
set kwCollection = ##class(%ArrayOfDataTypes).%New()
do kwCollection.SetAt(1, "a")
do kwCollection.SetAt(100, "b")
set kwDynamic = { "a": 1, "b": 100}

for kwArguments = kwList,kwCollection,kwDynamic {
    set sc = ##class(isc.py.Main).ExecuteFunction(random _ ".randint", ,kwArguments,,
.result)
    write result,!
}

set posList = $lb(1)
set kwDynamic = {"b": 100}
set sc = ##class(isc.py.Main).ExecuteFunction(random _ ".randint", posList, kwDynamic
,,.result)
write result,!

set posList = ##class(isc.py.util.Converter).EscapeStringList($lb("Positional: {0} {1
}! Keyword: {name}, {name2}", "Hello", "World"))
set kwDynamic = {"name":(##class(isc.py.util.Converter).EscapeString("Alice")),
                 "name2":(##class(isc.py.util.Converter).EscapeString("Bob"))}
```

```
set sc = ##class(isc.py.Main).ExecuteFunction("str.format", posList, kwDynamic,,.resu
lt)
write result,!
```

ExecuteFunctionArgs

ExecuteFunctionArgs method from isc.py.Main class. Signature:
- function - name of function to invoke. Can be nested, i.e. random.randint
- variable - name of python variable to write result to.
- serialization - how to serialize result
- result - write result into this variable
- args... - function arguments.

ExecuteFunctionArgs attempts to determine correct positional and keyword arguments from function signature (if available). It is recommended to call ExecuteFunction directly if ExecuteFunctionArgs is unable to construct a correct argument spec (and open an issue). Example:

```
set sc = ##class(isc.py.Main).ImportModule("random", ,.random)
set sc = ##class(isc.py.Main).ExecuteFunctionArgs(random _ ".randint", , ,.result, 1,
 100)
write result,!

set string = ##class(isc.py.util.Converter).EscapeString("Positional: {0}, {1}, {2},
{3}")
set arg1 = ##class(isc.py.util.Converter).EscapeString("Hello")
set arg2 = ##class(isc.py.util.Converter).EscapeString("World")
set arg3 = ##class(isc.py.util.Converter).EscapeString("Alice")
set arg4 = ##class(isc.py.util.Converter).EscapeString("Bob")
set sc = ##class(isc.py.Main).ExecuteFunctionArgs("str.format",,,.result, string, arg
1, arg2, arg3, arg4)
write result,!

set string = ##class(isc.py.util.Converter).EscapeString("Positional: {0} {1}! Keywor
d: {name}, {name2}")
set arg1 = ##class(isc.py.util.Converter).EscapeString("Hello")
set arg2 = ##class(isc.py.util.Converter).EscapeString("World")
set kwargs = "**" _ {"name":"Alice","name2":"Bob"}.%ToJSON()
set sc = ##class(isc.py.Main).ExecuteFunctionArgs("str.format",,, .result, string, ar
g1, arg2, kwargs)
write result,!
```

Summary

Python Gateway allows seamless integration between InterSystems IRIS and Python. Use it to add Python functionality to your InterSystems IRIS Application.

Links

- Python Gateway
- Python Gateway Samples
- Install Python 3.6.7 64 bit
- Python documentation/tutorial

Illustrated guide

There's also an illustrated guide in ML Toolkit user group. ML Toolkit user group is a private GitHub repository set up as part of the InterSystems corporate GitHub organization. It is addressed to the external users that are installing, learning or are already using ML Toolkit components including Python Gateway. To join ML Toolkit user group, please send a short e-mail at the following address: MLToolkit@intersystems.com and indicate in your e-mail the following details (needed for the group members to get to know and identify you during discussions):

- GitHub username
- Full Name (your first name followed by your last name in Latin script)
- Organization (you are working for, or you study at, or your home office)
- Position (your actual position in your organization, or “ Student ” , or “ Independent ”)
- Country (you are based in)

#Python #InterSystems IRIS
Check the related application on InterSystems Open Exchange