Article
Renato Banzai · Jul 28, 2022 6m read

Dealing with large content in Python Native API

Hi developers! In this article I ' m going to explore the use of Iris Python Native API in a specific problem: large strings to store.

Why Python Native API?

Python Native API for Intersystems IRIS offers an useful way to persist data that you can ' t have control over the schema or if the schema changes frequently. Combining the Native API with the IRIS Globals, you can easily use Iris Database as an Document Store. You can also see more details of globals use in documentation https://learning.intersystems.com/course/view.php?id=1110&ssoPass=1, or in my article https://community.intersystems.com/post/iris-python-suite-hitchhikers-guide-global-1 you can have an quick view of the use of globals.

The use case

In my article " Making a blog using Python + Iris Globals " I use Iris as the database that persist blog posts, as I wished that the post can would a flexible schema I persisted all posts inside an Iris Global. After testing this program I realize that each post was limited to the max size of a global content (today around 3MiB characters as mentioned in https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GCOStypes#GCOStypesstringslong) to compare I found that another " blog " like platform, the Wikipedia, has quite near max length of a content page: 2MiB. But what if the max length of a global didn ' t be enough to my use case? I can want to post a big post with a lot of text and scripts to make something… Now we get the point that makes this article title!

What documentation says

For large strings the docs recommends to break into parts of a global, like the example:

```
SET ^Data("Stream1",1) = "First part of stream...."
SET ^Data("Stream1",2) = "Second part of stream...."
SET ^Data("Stream1",3) = "Third part of stream...."
```

Now… we ride!!!

So, if you need to persist a big string with a length of more than the intersystems max string length limit you have some options:

- Control by yourself each time you have a big string coding specific for that global.
- Create a function using IRIS ObjectScript to control this to you
- **Create a function using Python to control this to you. (Which can be easier for the ones who used to program in Python and have no access direct to the Iris Server)**

Creating a function in Python to Store Large Strings into Globals

Now we have all the context we need to start creating a way to store large strings using Python. First we will configurate an environment to have our IRIS Multimodel Database for tests and development.

Requirements

- Docker: Install docker in your OS and pull the last version of InterSystems iris for community

```
docker pull store/intersystems/iris-community:2021.2.0.649.0
```

- Any text editor: you will need a text editor to create your python code.
- Python 3: I strong recommend to use the last stable version available to use.
- IRIS Native API (python): you can download the version that match your OS here: https://github.com/intersystems/quickstarts-python/tree/master/Solutions/nativeAPIwheel

Running an IRIS Community Server

If you didn ' t have an instance of Iris running to perform our experiment here, lets up a server using docker:

```
docker run --name my-iris -d --publish 9091:1972 --publish 9092:52773 store/intersyst
ems/iris-community:2021.2.0.649.0
```

This command will up an instance of IRIS Server, community version and you will need to change the password on your first access. The URL to access the management portal is:

http://localhost:9092/csp/sys/UtilHome.csp

login: SYSTEM

pass: sys

More options to run up a server with docker can be reached here: https://hub.docker.com//intersystems-iris-data-platform/plans/222f869e-567c-4928-b572-eb6a29706fbd?tab=instructions

Installing the Native API

After download the right version for you OS, install it using the command:

```
pip install <directory_that_you_downloaded_the_file>/irisnative-1.0.0-cp34-abi3-linux
_x86_64.whl
```

Note that you need to install the right wheel for your operational system. In my case I use linux, so I downloaded the linux version of the IRIS Native API so the command may variate for you.

Let ' s code

Now that we have an IRIS running, a Python 3 with the Native API properly installed, we can do what we

developers really enjoy, code!

First we import the libs that we will use, in this case only Iris Native will be enough, open the connection to our iris database and define a value that we want to work as the max length of each global.

```
import irisnative
connection_object = irisnative.createConnection("localhost",
                                                 9091,
                                                 "USER",
                                                 "_SYSTEM",
                                                 "your_new_password")

iris_object = irisnative.createIris(connection_object)
MAX_GLOBAL_LENGTH = 3641144
```

Note that I just put the configurations that can change mainly the yournewpassword.

Now we will define the function that receive an Iris Object, a value, a global name plus his subscripts.

```
def set_big_string(iris_object, value, *global_array):
    length = len(value)
    subscript = 0
    for x in range(0, length, MAX_GLOBAL_LENGTH):
        # adding a subscript to break the string
        global_list = list(global_array)
        global_list.append(subscript)
        global_tuple = tuple(global_list)

        partial_value = value[x:(x+MAX_GLOBAL_LENGTH)]
        subscript += 1
        iris_object.set(partial_value, *global_tuple)
    return
```

As you can see, the up code broke the value into smaller pieces and put into the same global but in the numeric subscripts created by the function starting by zero.

To read the global we make the inverse work, we make an iterator on the global itens and concatenate the string.

```
def get_big_string(iris_object, *global_array):
    node = iris_object.iterator(*global_array)
    res = ""
    for sub, val in node.items():
        res += val
    return res
```

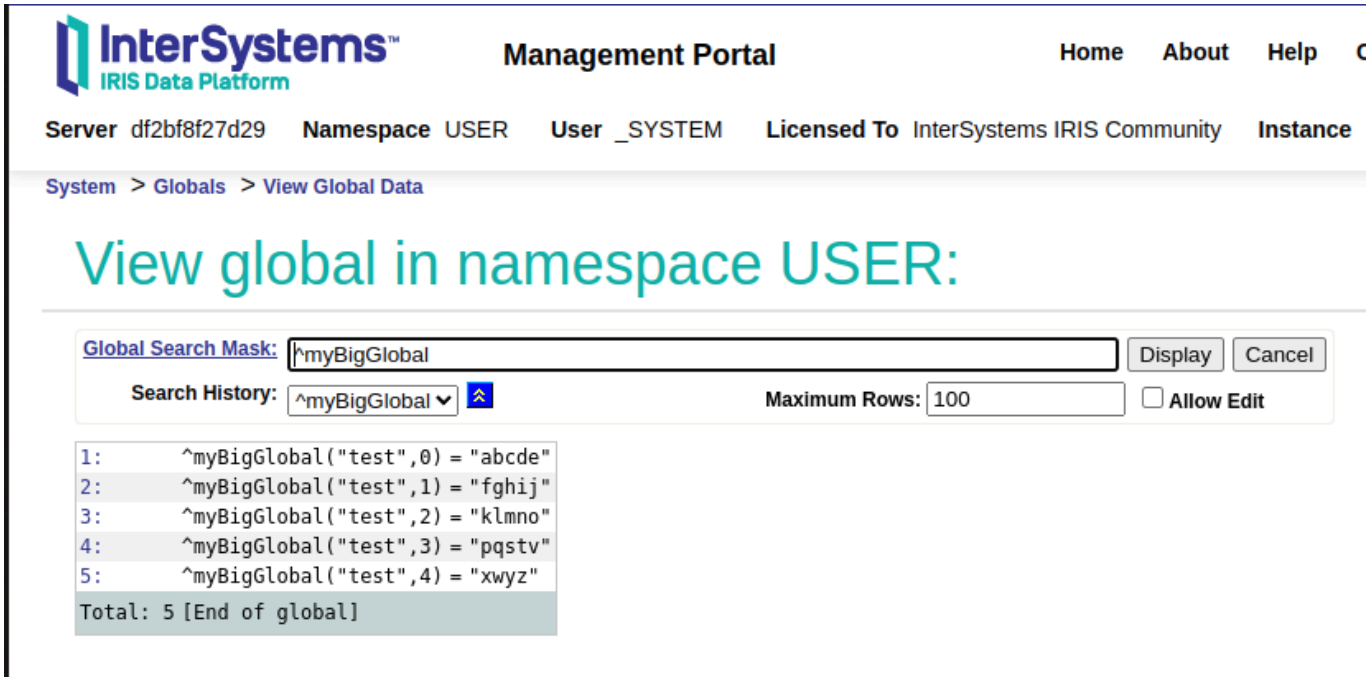
And to test to be more visual I reduce the number of the max global length, because I dont want to count more than 3 million characters =)

```
MAX_GLOBAL_LENGTH = 5
set_big_string(iris_object, "abcdefghijklmnopqstvxwyz", "myBigGlobal", "test")
print(get_big_string(iris_object, "myBigGlobal", "test"))
```

As result of the code we can see:

```
abcdefghijklmnopqstvxwyz
Process finished with exit code 0
```

And if we take a look at the Management Portal on the Global Viewer we can see how our code has stored our content.



Conclusion

Now we can see how easy for a python programmer can be interact with the IRIS Database. Only connecting the Native API to an Iris Database instance anyone can stay a few lines of programming to persist any data in a key value like database thats great! We can use thisto persist data in content management systems, make a document storage, configuration store (think in your yamls, jsons) and a lot of other challenges in non structured data that can show in our life. Comment if you like =)

Video in youtube to help understand the content

https://youtu.be/Bsu23vym8Ig

#Python #InterSystems IRIS

Source URL:https://community.intersystems.com/post/dealing-large-content-python-native-api