Article
Robert Cemper · Jul 15, 2021 4m read

The future position of ObjectScript

If you develop in IRIS you are confronted with two major phenomena:

- an incredibly fast and excellent designed data storage engine
- a scripting language to work on this storage engine, actually named ObjectScript

Storage engine

It is a hierarchically organized storage concept and was considered as
"old-fashioned" when "relational" was the Buzzword of the season.
Though - if using its strengths - it outperformes with its slim and efficient
construction all its "Sumo"-shaped competitors.
And this is still true and unbroken since day 1.
It's amazing to me how those competitors copied over time various features
of this storage engine indirectly confirming the quality of the basic concept.

ObjectScript

Looking at the language it is rather easy to separate the rather narrow set of language
elements that manipulate and navigate the storage engine that has seen just a few
extensions and some polishing over decades. I call this the **CORE.**
Its companions are **Navigation** (mostly similar to other languages)
and **Cosmetics** (everything to manipulate data content).

Navigation is an unavoidable structural requirement but has seen several additions
mostly for programmers' comfort. Not required but similar to newer languages.

Cosmetics is the fastest-growing field and until today its offers surprise me every
now and then. Its existence results from history where the Storage Engine was also
part of an operating system and nothing else on the HW.

History

From the past, developers were used to write applications with ObjectScript.
Even InterSystems did this on large scale (Interoperability aka Ensemble,
Analytics aka DeepSee, Health*...) ObjectScript was/is the "can all, does all"

But ObjectScript was rather a "Lonely Ranger". Attempts to have something more
popular on the market was BASIC. (Already a "dead horse" at that time.)
The chance to jump on the JavaScript wave was missed ~~1~~5 years ago.

The availability of a wide range of language adapters could hide the dimension
of the issue for some time, But it never reached the strength that the main
competitor showed when PL/SQL was frozen and replaced by Java.

Today

I see the number of qualified developers in ObjectScript shrinking over time by pure demoscopic effects. Finding and educating developers willing to write ObjectScript is a hard exercise that I experienced personally several times. And even if they are bright and highly skilled it is no guaranty that they will stay with you.
Demand is still rather dramatic.

In front of this background, I see the upcoming of Embedded Python as full-size language at the same level as ObjectScript as a major step forward with IRIS .
Navigation and Cosmetics are covered in public and adding the CORE functionalities doesn't require a dramatic learning step. So we see a much larger market of skilled development resources and the isolating uniqueness of ObjectScript and its limitations is broken.

Future

> With Python, I see a lot of new energy entering the world of IRIS.
After all, a step that some competitors did quite some time ago and this flaws of "never seen anywhere else" has been overcome.

> The paradigm: *"I write it, just because it is possible"* will be broken.
This releases us from a series of reinvented wheels.

> ObjectScript will still live in parallel and have a significant role in all activities close to the storage CORE. No one (hopefully) will use it for business logic or mimic interfaces that are available on a much larger scale outside already.

> ObjectScript will shrink to a dimension that we see today for C, C** or for code in whatever Assembly language. Existing applications will not be affected. The way Python was integrated seems to be a guaranty for a smooth migration without time pressure.

> Today's ObjectScript developers will not lose their jobs nor run out of demanding work. There are enough challenging tasks left around database design and maintenances.
But they could get rid of (in my eyes) unattractive tasks of CSS styles, coloring web pages, ...

> Not to forget the demanding challenge to read and understand existing code and interpret and communicate its content and logic. "GolfCode" may look like fun, but it is a deadly earnest business logic in many old traditional written applications and even some utilities in "%SYS".

> I see today's developers as the priests of ObjectScript similar to the Egyptian priests understanding hieroglyphs, reading "The Book of Dead" and celebrating their miracles.

July 15th, 2021

#Embedded Python #Languages #ObjectScript #Python #InterSystems IRIS