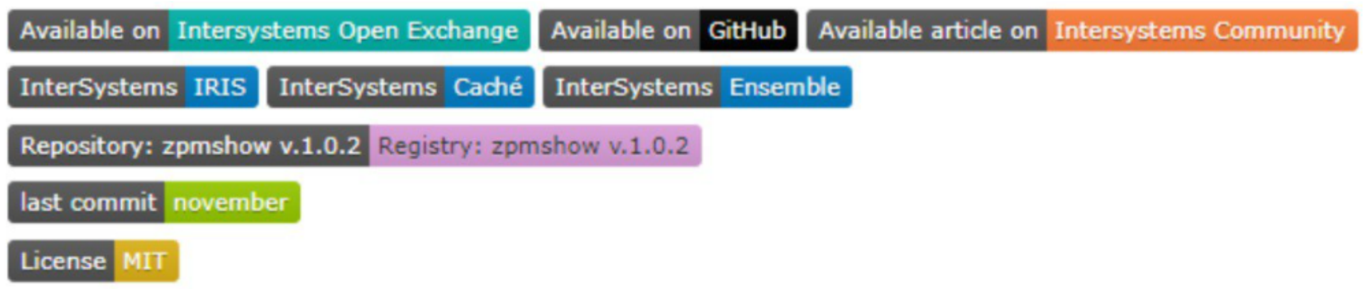
Article
Sergey Mikhailenko · Jan 18, 2022 5m read

Open Exchange

The practice of using badges for projects in OEX for a more convenient description of them

It is becoming more and more common to see beautiful badges in the README.MD file with useful information about the current project in the repositories of GitHub, GitLab and others.
For instance:


Available on Intersystems Open Exchange
Available on GitHub
Available article on Intersystems Community
InterSystems IRIS
InterSystems Caché
InterSystems Ensemble
Repository: zpmshow v.1.0.2
Registry: zpmshow v.1.0.2
last commit november
License MIT

The project is being developed The quality of the code, which also provides its own badge, which immediately shows the status of code validation of the project. If you insert a line into the README.MD file

```
 [![Quality Gate Status](https://community.objectscriptquality.com/api/project_badges
/measure?project=intersystems_iris_community%2Fappmsw-zpm-shields&metric=alert_status
)](https://community.objectscriptquality.com/dashboard?id=intersystems_iris_community
%2Fappmsw-zpm-shields)
```

And in the /.github/workflows/ directory github project the file objectscript-quality.yml, you can see this badge:

There are different services that grant these nameplates.
For example - Shield.io
It even executes screen forms, which simplify the creation of links, including for markup written in Markdown

I have been using a lot of different beautiful and useful nameplates in my projects for a long time.

As the project of the package manager ZPM matures, the requirements for the storage resources for package modules increase too.

It happens more and more often now that I need to know more detailed information, preferably observable at first glance at the first page, without opening the project files. Such data includes:
- what version of the project is stored in the repository. I need to see that without checking the module.xml file.
- how that version relates to the one in the public repository. Is it the right time already to update the release or not?...
- what ports are forwarded out in the settings of the docker file called dockerfile?

All of that is well performed by the shields.io service.

Show the version of the zpm project taken from the file module.xml

```
![Repo-GitHub](https://img.shields.io/badge/dynamic/xml?color=gold&label=GitHub%20mod
ule.xml&prefix=ver.&query=%2F%2FVersion&url=https%3A%2F%2Fraw.githubusercontent.com%2
Fsergeymi37%2Fzapm%2Fmaster%2Fmodule.xml)
```

You can complicate the link by adding the ability to click to open the corresponding file module.xml:

```
[![Repo-GitHub](https://img.shields.io/badge/dynamic/xml?color=gold&label=GitHub%20mo
dule.xml&prefix=ver.&query=%2F%2FVersion&url=https%3A%2F%2Fraw.githubusercontent.com%
2Fsergeymi37%2Fzapm%2Fmaster%2Fmodule.xml)](https://raw.githubusercontent.com/sergeym
i37/zapm/master/module.xml)
```

Show the version of the zpm project taken from the service

```
![OEX-zapm](https://img.shields.io/badge/dynamic/json?url=https:%2F%2Fpm.community.in
tersystems.com%2Fpackages%2Fzapm%2F&label=ZPM-
pm.community.intersystems.com&query=$.version&color=green&prefix=zapm)
```

Example of a link with a request for a service:

```
[![OEX-zapm](https://img.shields.io/badge/dynamic/json?url=https:%2F%2Fpm.community.i
ntersystems.com%2Fpackages%2Fzapm%2F&label=ZPM-pm.community.intersystems.com&query=$.
version&color=green&prefix=zapm)](https://pm.community.intersystems.com/packages/zapm
)
```

Show which ports are forwarded in the settings of the docker file called dockerfile

```
 ![Docker-ports](https://img.shields.io/badge/dynamic/yaml?color=blue&label=docker-co
mpose&prefix=ports%20-%20&query=%24.services.iris.ports&url=https%3A%2F%2Fraw.githubu
sercontent.com%2Fsergeymi37%2Fzapm%2Fmaster%2Fdocker-compose.yml)
```

For instance of a link with opening a file docker-compose.yml:

```
[![Docker-ports](https://img.shields.io/badge/dynamic/yaml?color=blue&label=docker-co
mpose&prefix=ports%20-%20&query=%24.services.iris.ports&url=https%3A%2F%2Fraw.githubu
sercontent.com%2Fsergeymi37%2Fzapm%2Fmaster%2Fdocker-compose.yml)](https://raw.github
usercontent.com/sergeymi37/zapm/master/docker-compose.yml)
```

However, when it comes to more complex metrics, their combinations, or projects inside a private local network, for those purposes I have decided to bring about my REST service, which shows the version of the ZPM module from the repository file and from the service https://pm.community.intersystems.com/

After installation, you will have a service zpm-shields to which you need to provide access without authentication.

Using these links you can get a svg file that can be inserted into README.MD for instance:

```
![Repo](http://localhost:52773/zpm-
shields/repo/mode?module=https:%2F%2Fgithub.com%2FSergeyMi37%2Fzapm&color=blue)
```

where the parameter values are:
zpm-shields/repo - extaction from the file module.xml version
module - link to the repository
color - for example #00987

```
![Registry](http://localhost:52773/zpm-shields/registry/mode?project=appmsw-
dbdeploy&color=gold)
```

where the parameter values are:
zpm-shields/registry - getting the version by request from the service
project - project name

```
![Repo+Registry](http://localhost:52773/zpm-shields/both/mode?module=sergeymi37%2Fapp
msw-dbdeploy&project=appmsw-dbdeploy&color=FFA07A)
```

where the parameter values are:
zpm-shields/both - extaction from the file module.xml version and getting the version by request from the service
project - project name
module - link to the repository

My service can also be used for local ZPM resources. To do that you need to utilise the full path of the local repository and private register.

I really like these badges. I think they might come in handy for you too.

#Best Practices #Docker #GitHub #InterSystems Package Manager (IPM) #InterSystems IRIS #Open Exchange
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/practice-using-badges-projects-oex-more-convenient-description-them