Article
Sergey Mikhailenko · Oct 20, 2020 11m read

Open Exchange

InterSystems: Solution for Technical Support and DBMS-Interoperability Administration

In this article, we'll talk about an application that I use every day when monitoring applications and integration solutions on the InterSystems IRIS platform and finding errors when they occur.

While looking for a solution for logging object changes in InterSystems IRIS, Ensemble, and Caché DBMS, I came across a great article about logging with macros. Inspired by the idea, I forked the project the paper had described and adapted it to some specific needs. The resulting solution is implemented as a panel subclass, %CSP.Util.Pane, which has the main window for commands, the Run button, and enabled command configuration.

This application enables viewing and editing global arrays, executing queries (including JDBC and ODBC), emailing search results as zipped XLS files, viewing and editing objects, as well as several simple graphs for system protocols.

The apptools-admin application is based on jQuery-UI, UiKit, chart.js, and jsgrid.js. You are welcome to have a look at the source code.

Installation

All installation methods are described in detail in the repo. However, the simplest approach is to use the package manager command:

```
zpm "install apptools-admin"

[apptools-admin]        Reload START
[apptools-admin]        Reload SUCCESS
[apptools-admin]        Module object refreshed.
[apptools-admin]        Validate START
[apptools-admin]        Validate SUCCESS
[apptools-admin]        Compile START
[apptools-admin]        Compile SUCCESS
[apptools-admin]        Activate START
[apptools-admin]        Configure START

http://hp-msw:52773/apptools/apptools.core.LogInfo.cls
http://hp-msw:52773/apptools/apptools.Tabs.PanelUikitPermissMatrx.cls?autoload=Matrix
[apptools-admin]        Configure SUCCESS
[apptools-admin]        Activate SUCCESS
```

The first suggested link must be opened in the address field of the browser. And in the loaded panel enter ? and press the "Execute" button. The application then displays command examples.

Commands

In the panel, you can run utilities, view and edit globals, and execute queries. Each launch is saved in history in the context of the namespace, so it can be found and repeated. In this context, the word "launch" means starting the execution of commands, and commands will mean everything that we enter in the panel. This screenshot shows an example of a global array ^%apptools.History view command

As you know, automatic error detection and notifications can be handled by popular solutions like Prometheus. But often the severity of errors can be assessed visually.

Very often I need to quickly get information about bugs in production in all namespaces.
For this, I implemented a utility:

##class(apptools.core.Production).FindAndDrawAllErr

This starts a daily search request for errors for each namespaces that contains working products, and allows you to view these errors with a quick transition to visual tracing. You can run this utility like any others in the apptools panel with the xec prefix.

All useful commands can be memorized in the global extensions, in the context of the scope, to be found and repeated at any time.

Globals

A large part of the apptools-admin application is dedicated to working with globals. Globals can be viewed in reverse order, as well as by applying a filter on both the link and the data. The displayed notes can be edited or deleted.

You can enter the * wildcard after the global name to get a list of globals with additional characteristics.

A second * will add a new field, Allocated MB.

A third one will add the Used MB field. This syntax resolves to a Union of the two reports, and the asterisks divide the report that is typically rather long into manageable sections.

When you get a report as a list of globals (in the screenshot above), you can follow the active links to view the global itself. You can also view and edit the global in the standard way from the management portal, by clicking R or W in the Permission field.

Quite often, writing to the global is used to log the states of variables and objects when debugging a project. I use special macros for this:

set $$$AppL("MSW","anyText")=$$$AppObJs(%request)

In this example, $$$AppL forms a link to a glob with the ^log prefix, and the date and time in the index value.

$$$AppObJs is the object serialization macro.

You can view the protocol global in the panel, and the object can be displayed in the window fully formatted.

Query

The function that sees almost as much use as globals is query. You run this function by entering a statement as a command.

For example, you can perform an SQL statement.

You can also save the result in the global ^mtempSQLGN.

Subsequently, the saved result in the global can be displayed in the panel.
![

Converting Reports to Excel Format

One of the things that was missing in the standard management portal was the ability to execute queries configured in the database JDBC or ODBC sources, output the results in XLS format, and then archive and send the file via email.

To achieve this in the application, you simply select the Upload to Excel file checkbox before executing the command.

This feature saves a lot of time in my daily routine, and allows me to successfully incorporate ready-made modules into new applications and integrated solutions.

To enable this functionality, you first need to configure the path for creating files on the server, user credentials, as well as the mail server. For that, in turn, you need to edit the nodes of the global program settings, ^%apptools.Setting.

Saving Reports Globally

Quite often, you need to save the results of a report execution to the global. For this, you can use these procedures:

	functions
For JDBC:	##class(apptools.core.sys).SqlToDSN
For ODBC:	##class(apptools.core.sys).SaveGateway
For SQL:	##class(apptools.core.sys).SaveSQL
For Query:	##class(apptools.core.sys).SaveQuery

For example, using the ##class(apptools.core.sys).SaveQuery function, saves the result of the query %SYSTEM.License:Counts to the global ^mtempGN.

You can then display in the panel what you've saved with the following command:

result ^mtempGN("%SYSTEM.License:Counts", 0)

https://lh5.googleusercontent.com/KCIekwZw3guq79GWxVdHYdAbWQc4u97-dr-hWT...

Enhanced Functionality Modules

What else simplified and automated my work? Changes that enabled me to execute custom modules when forming a query string. I can embed new functionality into the report on the fly — like active links for additional operations on the data. Let's see some examples.

We display the query result in the browser using the function:

##class(apptools.core.LogInfoPane).DrawSQL

Let's add the word marking function ##class(apptools.core.LogInfo).MarkRed to parameter 5.

In the same way, you can supplement the output with additional features, for example, active links or tooltips.

The globals editor in this solution is implemented according to the same principle.

Here's a list of functions for outputting globals and queries in tabular form:

	functions
For globals:	##class(apptools.core.LogInfoPane).DrawArray("^mtempSQLGN")
For SQL:	##class(apptools.core.LogInfoPane).DrawSQL("select * From %SYS.ProcessQuery")
For Query:	##class(apptools.core.LogInfoPane).DrawSQL("query %SYSTEM.License:Counts")
For global result:	##class(apptools.core.LogInfoPane).DrawSQL("result ^mtempSQLGN")

Working with the apptools.core.Parameter Class
This link will open the CSP application in a browser in the context of an instance on which apptools-admin is installed:

http://localhost:52773/apptools/apptools.Form.Exp.cls?NSP=APP&SelClass=apptools.core.Parameter

Or select the active link in the panel.

The CSP application will be loaded for editing instances of stored classes, in this example: apptools.core.Parameter.

Creating a apptools.core.Parameter via the Table Navigator

If you open this link in a browser in the context of the instance on which apptools-admin is installed:

http://localhost:52773/apptools/apptools.Form.Exp.cls?panel=AccordionExp&NSP=APP

Or select the active link in the panel.

The CSP application will be loaded for navigating the stored classes with the ability to edit them.

An example of a simple CSP application

If you open this link in a browser in the context of the instance on which apptools-admin is installed:

http://localhost:52773/apptools/apptools.Tabs.PanelSample.cls

Or select the active link in the panel.

This example also shows the ability to edit class instances apptools.core.Parameter.

Graphs

To visualize database growth, the application offers a page that displays a graph of the monthly measured database size. This graph is derived from the IRIS file.log (cconsole.log for Caché) on records "Expand" retrospectively from the current day.

The program traverses the protocol, finds the database extension records and subtracts the incremental megabytes from the current database size. It turns out a graph of the growth of databases.

For example, the screenshot below shows a graph of events in InterSystems IRIS formed by the protocol file.

Another example below: a schedule of events in the system based on the system protocol file.log (cconsole.log).

Summary

The application we've discussed in this article was designed to help me perform my daily tasks. It includes a set of modules which you can use as building blocks for a custom administrator tool. I would be very glad if you found it useful in your work. You are welcome to add your wishes and suggestions as tasks to the project repo.

#Tips & Tricks #Tools #Ensemble #InterSystems IRIS #Open Exchange
Check the related application on InterSystems Open Exchange