Article
Oleg Dmitrovich · Jan 4, 2018 5m read

Caché audit & DeepSee

Apart from the database server itself, the standard bundle of the Caché DBMS includes **DeepSee**, a real-time business intelligence tool. DeepSee is the quickest and the simplest way of adding OLAP functionality to your Caché application.

Another standard component is an **Audit** subsystem with a web interface, which has the options for expanding with your own event types and an API for using in an application code.

Below is a **small example of the joint use of these subsystems** that answers the following questions: **who** did **what** and **when** in an information system?

Audit subsystem

This subsystem is intended for registering events occurring in the system. The Caché management portal comes with a ready web interface supporting search, filtering, export, record removal and etc.

Audit functions can be accessed from the application code via an API, where Security.Events class are used for registering types of events and %SYS.Audit class for registering events themselves. %SYSTEM.Security:Audit() is a convenient method for registering events. By default, the audit subsystem is inactive, but can be started from: Management portal — System administration — Security — Audit.

Below is a sample page with several buttons. Clicks on these buttons are registered in the audit database.

An example of using audit in an application code

```
/// Register your own event types
ClassMethod EventTypesRegister() As %Status {
    #; Event types register in %SYS namespace
    new $namespace s $namespace = "%SYS"
    #; set sc = ##class(Security.Events).Create( Source, Type, Name, Description )
    s src = "habra", type="audit"
    s scC = ##class(Security.Events).Create(src, type, "create", "Create event exampl
e")
    s scR = ##class(Security.Events).Create(src, type, "read", "Read event example")
    s scU = ##class(Security.Events).Create(src, type, "update", "Update event exampl
e")
    s scD = ##class(Security.Events).Create(src, type, "delete", "Delete event exampl
e")
    Q scC && scR && scU && scD
}
/// Simple html form with few buttons
```

```
ClassMethod OnPage() As %Status {
    &html<<!DOCTYPE html><html><head><title>Audit Example</title></head><body>
    <h3>Hello, #($username)#!</h3><hr>
    <form method='post'
    style='display:flex;justify-content:space-around;align-items:center;'>
    <button name='create'>Create</button>
    <button name='read'  >Read </button>
    <button name='update'>Update</button>
    <button name='delete'>Delete</button>
    </form>
    </body></html>>
    Quit 1
}
/// Parse html form submit
ClassMethod OnPreHTTP() As %Boolean [ ServerOnly = 1 ] {
    #; form send to server only button name, like below:
    #; http://[server]/[app]/[class]?name=
    s name = $order( %request.Data("") ) ;which button?
    #; save button name into Audit subsystem (Source,Type,Name,EventData,Description
)
    s sc = ##class(%SYSTEM.Security).Audit("habra","audit",name,"button","1984")
    Q 1
}
```

DeepSee business analysis technology

DeepSee includes a variety of tools for data storage creation, data analysis and visualization, a user portal, reports, printing support, export capabilities and so on. It uses Caché security and audit subsystems. There are many ways of integrating DeepSee with with almost any application, from a terminal to a web app.

Another useful feature of DeepSee is the possibility to analyse operational data. This is achieved through background synchronization of the cube with your application data. For this purpose, DSTIME and DSINTERVAL parameters are defined in the data class. When compiling the class, Caché generates an additional change registration code. When synchronization is started, only a limited part of the cube ’ s data is updated, and synchronization can be performed immediately after changes that occur in OLTP classes, which allows us to talk about “ real-time ” business analysis.

To use background cube updates in our example, you need to override **DSTIME** and **DSINTERVAL** parameters to the %SYS.Audit system class and compile it.

Changes in %SYS.Audit:

```
------------------End Documentation---------------------------------
*/
Parameter DSTIME = "AUTO";
Parameter DSINTERVAL = 5;
```

All together

After the preliminary configuration of the area (you need to enable the mapping of the ^CacheAuditD global) and the corresponding setting of the web application for working with DeepSee, we can proceed to defining the cube.

Let ’ s specify the source data class. The properties of the class will become the basis for defining the dimensions of the cube. Username, Event and UTCTimeStamp properties of the %SYS.Audit store information for answering the questions, set at the beginning of the article. Let ’ s use the Username property to define the **Who** dimension, the Event property for **What**, and UTCTimeStamp for **When**. The number of records is used as the default measure.

After compilation and the initial population of the cube, you need to configure various data views in DeepSee Analyzer (pivot tables). They, in turn, become data sources for visual components - widgets. Widgets are grouped into dashboards where users get access to. The user portal allows you to organize the work of system users with dashboards without writing a single line of code.

Analysis of the audit log can easily show, for instance, that user A is more prone to deleting data than user B. In a real system with a large number of events and users, this information might remain unnoticed for someone browsing the audit log. Including additional data to the cube may reveal a lot more “ interesting ” information about users activities in the system. Being able to ask the right questions is the key ability here.

Useful links:
More about DeepSee
Example of source code
DeepSee Web

#Security #Caché