Anuncio
Luis Angel Pére... · 21 oct, 2022

glsdb: Objetos JavaScript que son en realidad Objetos IRIS

Quiero anunciar el lanzamiento de algo muy interesante - y revolucionario, de hecho. Puede sonar exagerado, pero no creo que hayáis visto nada como esto, ¡ni si quiera imaginar que sería posible!

Hemos sacado un nuevo módulo JavaScript/Node.js llamado *glsdb* del que podéis leer todo aquí:

https://github.com/robtweed/glsdb

No obstante, para el propósito de este anuncio, me voy a centrar en una parte de *glsdb*: sus APIs que abstraen las Clases de IRIS (o Cache) como Objetos JavaScript equivalentes.

Con esto quiero decir que los Objetos de JavaScript serán en realidad ¡Objetos IRIS persistidos en la base de datos!

Suponed que tengo instalados los datos SAMPLES del repositorio de InterSystems, como se describe aquí:

https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=ASAMPLES

Así que, para acceder a estos datos SAMPLES usando *glsdb*, primero tengo que abrir una conexión con IRIS (*glsdb* usa nuestro interfaz *mg-dbx* para hacer esa conexión), por ejemplo:

```
const {glsDB} = await import('glsdb');
let glsdb = new glsDB('iris');
let options = {
  connection: 'network',
  host: '172.17.0.2',
  tcp_port: 7042,
  username: "_SYSTEM",
  password: "xxxxxx",
  namespace: "SAMPLES"
}

glsdb.open(options);
```

Ahora, si deseo acceder a un registro de Empleado (*Employee*) en la base de datos SAMPLES, bastaría con lo siguiente:

```
let Employee = glsdb.irisClass('Sample.Employee');
let employee = Employee(101);
```

Por ahora, quizás, no os sorprenda.

Pero aquí es donde empieza lo alucinante. Hemos creado un Objeto JavaScript llamado *employee* que parece representar la instancia Employee con un OID de 101, pero realmente es un Objeto JavaScript bastante especial: es un Objeto Proxy de JavaScript que está directamente mapeado a la instancia física de la Clase IRIS en la base de datos. Tenemos acceso directo a todas las propiedades y métodos de la Clase IRIS así que, cuando manipulamos el Objeto Proxy, ¡estamos realmente manipulando la instancia de la Clase IRIS en la base de datos! Por lo tanto, podremos hacer cosas como esta:

```
let salary = employee.Salary;
console.log('employee.Salary = ' + salary);
let company = employee.Company;
let name = company.Name;
console.log('Company: ' + name);
```

Lo importante que hay que tener en cuenta aquí es que el objeto *employee* no es una copia de JavaScript almacenada en memoria de una instancia de la Clase IRIS - es la instancia de la Clase IRIS ***real*, en la base de datos**!

Como podéis ver aquí debajo, podemos acceder a atributos de subclases a través de la relación entre el Empleado (*Employee*) y la Compañía (*Company*) como se haría en ObjectScript, pudiendo hacerlo de una sóla vez:

```
console.log(employee.Company.Name);
```

De tal forma podremos acceder, a través de las propiedades, directamente a la Clase física de IRIS en la base de datos.

También podremos guardar los cambios de los objetos - para ello utilizaremos un método especial llamado *save()* , por ejemplo: *employee.save()*

Para crear una nueva instancia de un empleado (sin especificar el OID), por ejemplo:

```
let newEmployee = Employee();
```

glsdb proporciona un método más rápido para definir y guardar un nuevo registro de una sola vez, este método será *set()* , por ejemplo:

```
let ok = newEmployee._set({
    Title: 'Manager'
    Salary: 60000
})
```

Podréis incluso examinar los métodos y propiedades disponibles para la Clase:

```
console.log(employee._properties);
console.log(employee._methods);
```

Esto ha sido un vistazo rápido a *glsdb*: ¡espero que os haya permitido conocer un poco lo fantástico que es!

Es posible que os estéis preguntando cómo se pueden hacer este tipo de cosas.

A modo de resumen, todo se reduce a una funcionalidad relativamente nueva de JavaScript: Objetos Proxy. Los Objetos Proxy son objetos especiales que actúan, como su nombre indica, como proxy de otro objeto real. Pero su verdadero poder está en el hecho de que puedes configurar pasarelas para acceder o cambiar las propiedades del Objeto Proxy y (esta es la parte importante) puedes aplicar lógica personalizada a esas pasarelas. En el caso de *glsdb*, esa lógica personalizada utiliza las APIs *mg-dbx* por detrás para llevar a cabo el acceso y los cambios equivalentes que sucedan sobre el Objeto Proxy en la clase IRIS correspondiente.

De todas formas, esto es un vistazo rápido al increible mundo de posibilidades con *glsdb.* Lo que he publicado es una primera versión, por lo que estoy seguro de que hay aspectos del proxy de IRIS que irán siendo modificados. Probadlo y ved lo que puede hacer, si encontráis algo que no funciona correctamente no dudéis en poneros en

contacto conmigo.

Y por cierto, comprobad las otras APIs que *glsdb* proporciona - ¡son igual de interesantes y alucinantes!

Por último, *glsdb* es software Open Source, así que podéis probar el código y postearlo en el repositorio de Github si queréis echar una mano con los desarrollos futuros.

#JavaScript #Modelo de datos de objetos #Node.js #Caché #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/glsdb-objetos-javascript-que-son-en-realidad-objetos-iris