Anúncio
Cristiano Silva · Out. 25, 2022

glsdb: Objetos JavaScript que são na realidade Objetos IRIS

Eu gostaria de anunciar o lançamento de algo realmente bastante interessante - revolucionário na verdade. Isso pode soar exagerado, mas acho que você não viu nada parecido com isso, ou mesmo pensou que fosse possível!

Lançamos um novo módulo JavaScript/Node.js chamado *glsdb*, mais informações:

https://github.com/robtweed/glsdb

No entanto, para os propósitos deste anúncio aqui, quero apenas focar em uma parte do *glsdb*: suas APIs que abstraem classes IRIS (ou Caché) como objetos JavaScript equivalentes.

Vamos dar uma olhada rápida no que quero dizer com isso, porque quero dizer Objetos JavaScript que na verdade são Objetos IRIS que residem no banco de dados!

Suponha que eu instalei os dados SAMPLES do repositório InterSystems, por exemplo, conforme descrito aqui:

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

Então, para acessar esses dados SAMPLES usando o *glsdb*, primeiro preciso abrir uma conexão com o IRIS (o *glsdb* usa nossa interface *mg-dbx* para fazer essa conexão), por exemplo:

```
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);
```

Agora que fiz isso, posso fazer o seguinte para acessar um registro de Funcionário (Employee) no banco de dados SAMPLES:

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

Até agora, tão surpreendente, talvez.

Mas é aqui que começa a diversão alucinante. Criamos um objeto JavaScript chamado *employee* que parece representar uma instância de Funcionário (Employee) com o OID 101, mas na verdade é um objeto JavaScript bastante especial: é um JavaScript Proxy Object que é mapeado diretamente para a instância física da classe IRIS no banco de dados. Eu tenho acesso direto a todas as propriedades e métodos da Classe IRIS, e quando eu manipulo o Proxy Object, na verdade estou manipulando a instância da Classe IRIS no banco de dados! Então agora eu posso fazer coisas como essa:

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

O importante a ser percebido aqui é que meu objeto *employee* não é uma cópia JavaScript na memória da instância da Classe IRIS - é a instância ***real*** da Classe Class, no **banco de dados**!

E como você pode ver acima, posso até encadear o relacionamento entre o Funcionário (Employee) e a Empresa (Company) como você faria em ObjectScript. Na verdade, eu posso até fazer isso de uma só vez:

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

e encadeie diretamente pelas propriedades, diretamente na Classe Física IRIS no banco de dados!

Eu também posso salvar as alterações no objeto - para isso eu uso um método especial chamado save(), ex.: *employee.save()*

Para criar uma nova instância de um funcionário, eu simplesmente não especifico um OID, ex.:

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

glsdb fornece um atalho para definir e salvar um novo registro de uma só vez, usando o método *set()* , ex.:

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

Você pode até mesmo inspecionar os métodos e propriedades disponíveis da Classe:

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

Então, essa é uma visão geral do *glsdb*: espero que dê um gostinho de como é insano!

Você provavelmente está se perguntando como é possível fazer esse tipo de coisa.

Bem, em resumo, tudo se resume a um recurso relativamente novo do JavaScript: objetos proxy. Objetos Proxy são objetos especiais que atuam, como o próprio nome indica, como um proxy para outro objeto real. Mas o verdadeiro poder deles está no fato de que você pode definir "armadilhas" para acessar ou alterar as propriedades do Objeto Proxy e (e essa é a chave) você pode aplicar lógica personalizada a essas armadilhas. No caso do *glsdb*, essa lógica personalizada usa as APIs *mg-dbx* nos bastidores para realizar o acesso equivalente ou a alteração que está acontecendo no objeto Proxy para uma classe IRIS correspondente.

De qualquer forma, esse é um vislumbre rápido do mundo louco possível com o *glsdb*! O que eu publiquei é uma versão antecipada, então tenho certeza de que há aspectos do proxy IRIS que precisarão de ajustes. Se você quiser experimentá-lo e ver o que ele pode fazer, por favor, avise-me se você encontrar algo que não funcione adequadamente.

Ah, e a propósito, confira as outras APIs que o *glsdb* fornece - elas são igualmente interessantes e alucinantes!

E, finalmente, *glsdb* é um software de código aberto, então, por favor, brinque com o código e poste PRs no repositório do Github se você quiser ajudar no desenvolvimento posterior.

#JavaScript #Modelo de Dados Objeto #Node.js #Caché #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/glsdb-objetos-javascript-que-s%C3%A3o-na-realidade-objetos-iris