Artículo
Luis Angel Pére... · 22 nov, 2022 Lectura de 5 min

InterSystems IRIS con Laravel (a través de ODBC)

Últimamente me han preguntado en varias ocasiones cómo hacer que el Framework Laravel funcione con InterSystems IRIS Data Platform. Ha pasado un tiempo desde la publicación de este anuncio sobre Laravel e InterSystems Caché. En este artículo mostraremos unas breves instrucciones sobre cómo configurar un proyecto Laravel para usarlo con InterSystems IRIS a través de ODBC.

¿Qué es Laravel?

Laravel es un *framework* PHP basado en arquitectura MVC. Usar Laravel simplifica y acelera el desarrollo del *backend*, a la vez que permite crear aplicaciones web modernas y seguras. Es muy fácil de aprender, muy utilizado en el mundo PHP y suele ser uno de los *frameworks* de *backend* más populares, según el ranking de github.com, que se puede ver en este vídeo. Claramente podemos ver los beneficios de combinar estas capacidades con la flexibilidad y el rendimiento de InterSystems IRIS como base de datos.

Esta publicación está organizada en 4 pasos, que representa una secuencia de acciones que hay que completar para que funcione la conexión. La forma de completar cada paso puede variar dependiendo de la plataforma. Los comandos se muestran para Ubuntu 22.02 (x64).

Configuración del Driver Manager (unixODBC)

Para hacer que la conexión funcione, necesitamos instalar un Driver Manager. Los driver managers más utilizados son 'unixODBC' and 'iODBC'. Utilizaremos 'unixODBC', que se puede descargar aquí http://www.unixodbc.org/ . Id a la sección 'Download' para descargar y ejecutar el build. Alternativamente, las instrucciones de build también se pueden encontrar aquí. Aquí utilizaremos paquetes del gestor de paquetes 'apt' para Ubuntu.

Instalación de paquetes

Instalamos el paquete unixodbc acompañado por libodbccr2 que proporciona la librería unixODBC Cursor.

```
sudo apt update
sudo apt -y install unixodbc libodbccr2 odbcinst
```

Creación de un enlace para Cursor Library
En algunos casos puede haber problemas con dependencias compartidas después de la instalación de unixODBC. Esto aparece como el error 'Can't open cursor lib'. En internet encontramos varias soluciones alternativas. Para resolver este problema, hacemos un enlace simbólico a una librería que queramos. Primero, localizamos la librería:

```
sudo find / -type f -name "libodbccr*"
```

Y después creamos un enlace

```
sudo ln -s /usr/lib/x86_64-linux-gnu/libodbccr.so.2.0.0 /etc/libodbccr.so
```

Configuración del Driver ODBC para InterSystems IRIS

El driver ODBC para InterSystems IRIS se puede obtener de varias maneras. Por ejemplo, está incluido en todos los kits de InterSystems IRIS. Otra opción sería el Portal de Distribución en wrc.intersystems.com.

También, los *drivers* para todas las plataformas soportadas se pueden encontrar aquí: https://intersystems-community.github.io/iris-driver-distribution/

Descargad, desempaquetad e instalad el Driver ODBC:

```
sudo mkdir -p /usr/lib/intersystems/odbc
cd /usr/lib/intersystems/odbc/

sudo wget -q https://raw.githubusercontent.com/intersystems-community/iris-driver-distribution/main/ODBC/lnxubuntu2004/ODBC-2022.1.0.209.0-lnxubuntu2004x64.tar.gz
sudo tar -xzvf /usr/lib/intersystems/odbc/ODBC-2022.1.0.209.0-lnxubuntu2004x64.tar.gz
sudo ./ODBCinstall

sudo rm -f ODBC-2022.1.0.209.0-lnxubuntu2004x64.tar.gz
```

Después, el *driver* se ubicará en esta carpeta /usr/lib/intersystems/odbc/bin/.

Podéis encontrar información adicional sobre los *drivers* y su uso en la documentación. Aquí utilizamos libirisodbcur6435.so como una librería de drivers.

Configuración del proyecto Laravel

La forma tradicional y práctica de interactuar con una base de datos desde Laravel sería usar su paquete Eloquent . Eloquent es un mapeador relacional de objetos (un ORM de toda la vida) que se incluye por defecto dentro del *framework* Laravel. Solo unos pocos proveedores de DBMS están soportados fuera de la herramienta. Así que para implementar la conexión y las especificidades de la construcción de consultas SQL para InterSystems IRIS (via ODBC) hace falta escribir código PHP adicional. Gracias a @Jean Dormehl que descubrió y solucionó esta brecha en InterSystems Caché y que también se puede usar para InterSystems IRIS.

En este artículo describimos los pasos para configurar la conexión para el proyecto Laravel existente, usando el paquete jeandormehl/laracache, asumiendo que la instalación y configuración de php, composer y Laravel ya se ha realizado.

Instalación de php-odbc
Aseguraos de que el módulo php-odbc está instalado. Podéis comprobar la lista de módulos instalados con el comando:

```
php -m | grep odbc
```

Para instalar la extensión php-odbc, usad el siguiente comando utilizando la versión adecuada

de php instalada en vuestro entorno

```
sudo apt -y install php8.1-odbc
```

Configuración del paquete 'jeandormehl/laracache'
Id a vuestro directorio del proyecto Laravel, instalad el paquete y publicad su archivo de configuración.

```
composer require jeandormehl/laracache
php artisan vendor:publish --tag=isc
```

Configuración de la conexión con IRIS

Editad vuestro fichero .env para que contenga las configuraciones necesarias para conectar a una base de datos. Para los usuarios de Unix debería ser algo tal que así:

```
DB_CONNECTION=isc
DB_WIN_DSN=
DB_UNIX_DRIVER=/usr/lib/intersystems/odbc/bin/libirisodbcur6435.so
DB_HOST=127.0.0.1
DB_PORT=1972
DB_DATABASE=USER
DB_USERNAME=_SYSTEM
DB_PASSWORD=sys
```

Después de editar el archivo .env puede que sea útil limpiar la caché de configuración de aplicaciones:

```
php artisan config:cache
```

Uso

Vamos a intentar recuperar algunos datos usando nuestro nuevo paquete. Como ejemplo, crearemos un Model heredado de Laracache\Cache\Eloquent\Model . Solo para probar, contaremos el número de mensajes enviados en el namespace de Interoperabilidad habilitado.

```
nano app/Models/EnsMessageHeader.php
```

```
<?php
namespace App\Models;

use Laracache\Cache\Eloquent\Model;

class EnsMessageHeader extends Model
{
    protected $table = 'Ens.MessageHeader';
    protected $fillable = [
        'MessageBodyClassName'
    ];
}
```

Para ejecutar una consulta, podemos crear un comando de consola Artisan como este:

```
nano routes/console.php
```

```
<?php
use Illuminate\Foundation\Inspiring;
use Illuminate\Support\Facades\Artisan;
use App\Models\EnsMessageHeader;

Artisan::command('iris:test', function () {
    echo EnsMessageHeader::count() . PHP_EOL;
});
```

Después, al ejecutar el siguiente comando, deberíamos recuperar el número de registros

```
php artisan iris:test
```

Este escenario debería funcionar en un amplio rango de productos basados en InterSystems IRIS.

#ODBC #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/intersystems-iris-con-laravel-trav%C3%A9s-de-odbc