TP N 7

Redis

ISTA TINGHIR
Realisé par: Groupe7
Ayoub Hafid
Yaala Salaheddine
Faska Tariq

le: 19 mai 2023

Module: M205
Developper en Back-End

❖ 1. introduction.

Redis est une base de données avancée de type clé-valeur, en open source. On le qualifie souvent de serveur de structure de données, car les clés peuvent contenir des chaînes de caractères, des hachages, des listes, des ensembles et des ensembles triés.
Avant d'utiliser Redis avec Laravel, nous vous encourageons à installer et utiliser l'extension PHP phpredis via PECL. Cette extension est plus complexe à installer par rapport aux packages PHP "utilisateurs", mais peut offrir de meilleures performances pour les applications qui utilisent intensivement Redis..
Si vous ne parvenez pas à installer l'extension phpredis, vous pouvez installer le package predis/predis via Composer. Predis est un client Redis entièrement écrit en PHP et ne nécessite aucune extension supplémentaire.

```
composer require predis/predis
```

Configuration

Vous pouvez configurer les paramètres Redis de votre application via le fichier de configuration config/database.php. À l'intérieur de ce fichier, vous trouverez un tableau "redis" contenant les serveurs Redis utilisés par votre application :

```
'redis' => [

    'client' => env('REDIS_CLIENT', 'phpredis'),

    'default' => [
        'host' => env('REDIS_HOST', '127.0.0.1'),
        'password' => env('REDIS_PASSWORD'),
        'port' => env('REDIS_PORT', 6379),
        'database' => env('REDIS_DB', 0),
    ],

    'cache' => [
        'host' => env('REDIS_HOST', '127.0.0.1'),
        'password' => env('REDIS_PASSWORD'),
        'port' => env('REDIS_PORT', 6379),
        'database' => env('REDIS_CACHE_DB', 1),
    ],

],
```

Predis

Si vous souhaitez que votre application interagisse avec Redis via le package Predis, vous devez vous assurer que la valeur de la variable d'environnement REDIS_CLIENT est définie sur predis:

```
'redis' => [

    'client' => env('REDIS_CLIENT', 'predis'),

    // ...
],
```

Interacting With Redis

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redis;

class CustomerController extends Controller
{
    //

    public function index()
    {

        $custumers = Redis::keys('name_*');
        return view('user.profile',['custumers'=>$custumers]);

    }
}

```

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <table border="1">
        <tr><td>Customer Name</td><td>Action</td></tr>
        @foreach($custumers as $custumer)
        <tr><td>{{$custumer}}</td><td></td></tr>
        @endforeach
    </table>
</body>
</html>
```

Vous pouvez appeler n'importe quelle commande Redis sur la façade Redis. Laravel utilise des méthodes magiques pour transmettre les commandes au serveur Redis. Si une commande Redis attend des arguments, vous devriez les passer à la méthode correspondante de la façade :

```php
use Illuminate\Support\Facades\Redis;

Redis::set('name', 'Taylor');

$values = Redis::lrange('names', 5, 10);
```