Client Redis pour PHP

Pour utiliser la bibliothèque `"predis/predis": "1.1"` en PHP, vous faites référence à un client Redis pour PHP qui peut être installé via Composer. Voici un guide étape par étape pour l'installer et l'utiliser dans votre projet PHP :

---

1. Installer Composer (si ce n'est pas déjà fait)

Si vous n'avez pas Composer installé, téléchargez-le et installez-le depuis getcomposer.org. C'est un gestionnaire de dépendances pour PHP.

---

2. Configurer Votre Projet

Si vous n'avez pas encore de projet, créez un répertoire pour celui-ci :

```
mkdir my-redis-project
cd my-redis-project
```

---

3. Ajouter Predis à Votre Projet

Exécutez la commande Composer suivante pour installer la version 1.1 de Predis :

```
composer require predis/predis:1.1
```

Cela téléchargera la version 1.1 de Predis et générera un fichier `composer.json` (ainsi qu'un répertoire `vendor` avec la bibliothèque).

Alternativement, si vous avez déjà un fichier `composer.json`, vous pouvez y ajouter manuellement cette ligne :

```
{
    "require": {
        "predis/predis": "1.1"
    }
}
```

Ensuite, exécutez :

```
composer install
```

---

4. Inclure l'Autoloader

Dans votre script PHP, incluez l'autoloader de Composer pour charger Predis :

```php
require 'vendor/autoload.php';
```

---

5. Exemple d'Utilisation de Base

Voici un exemple simple de la manière de se connecter à un serveur Redis et d'utiliser Predis :

```php
<?php
require 'vendor/autoload.php';

use Predis\Client;

// Créer une nouvelle instance de client Redis
$redis = new Client([
    'scheme' => 'tcp',
    'host'   => '127.0.0.1', // Remplacez par l'hôte de votre serveur Redis
    'port'   => 6379,        // Port Redis par défaut
]);

// Définir une paire clé-valeur
$redis->set('mykey', 'Hello, Redis!');

// Récupérer la valeur
$value = $redis->get('mykey');
echo $value; // Affiche : Hello, Redis!

// Exemple avec une liste
$redis->lpush('mylist', 'item1');
$redis->lpush('mylist', 'item2');
$list = $redis->lrange('mylist', 0, -1);
```

```php
print_r($list); // Affiche : Array ( [0] => item2 [1] => item1 )
?>
```

---

6. Options de Configuration

Vous pouvez personnaliser la connexion en passant un tableau d'options au constructeur `Client`. Par exemple :

```php
$redis = new Client([
    'scheme' => 'tcp',
    'host'   => 'redis.example.com',
    'port'   => 6379,
    'password' => 'yourpassword', // Si l'authentification est requise
    'database' => 0,              // Sélectionner une base de données Redis spécifique
]);
```

---

7. Commandes Courantes

Predis prend en charge la plupart des commandes Redis directement sous forme de méthodes. Voici quelques exemples : - **Chaînes** : `set`, `get`, `incr`, `decr` - **Listes** : `lpush`, `rpush`, `lpop`, `rpop`, `lrange` - **Ensembles** : `sadd`, `smembers`, `srem` - **Hashes** : `hset`, `hget`, `hgetall`

Pour une liste complète, consultez la documentation Predis ou la référence des commandes Redis.

---

8. Gestion des Erreurs

Enveloppez vos appels Redis dans un bloc try-catch pour gérer les erreurs de connexion ou d'exécution :

```php
try {
    $redis->set('test', 'value');
    echo $redis->get('test');
} catch (Exception $e) {
    echo "Erreur : " . $e->getMessage();
}
```

---

9. Prérequis

- Assurez-vous d'avoir un serveur Redis en cours d'exécution (par exemple, localement via `redis-server` ou sur un hôte distant).
- PHP 5.3.0 ou supérieur (Predis 1.1 est compatible avec les versions plus anciennes de PHP, mais vérifiez votre environnement).

C'est tout ! Vous êtes maintenant prêt à utiliser Predis 1.1 dans votre projet PHP. Faites-le moi savoir si vous avez besoin d'aide pour des opérations Redis spécifiques ou pour le dépannage !