ip_addresses

Net and Light

2023 - 2024

Résumé

Cette documentation décrit le script Python utilisé pour extraire les dernières adresses IP des châssis à partir de la base de données MySQL. Le script lit les configurations à partir d'un fichier JSON et utilise ces informations pour se connecter à la base de données et récupérer les adresses IP.

Table des matières

1 Introduction

Cette documentation présente le script Python "ip_addresses.py" conçu pour se connecter à une base de données MySQL, extraire les dernières adresses IP des châssis et les retourner sous forme de dictionnaire. Le script utilise le module `mysql.connector` pour la connexion à la base de données et le module `json` pour lire les configurations nécessaires.

2 Contenu du Script Python

2.1 Structure générale

Le script Python exécute les étapes suivantes :

- **Importation des modules** : Importe les modules nécessaires pour la connexion à la base de données et la gestion des fichiers JSON.
- **Lecture du fichier de configuration** : Lit les configurations de connexion à la base de données depuis un fichier JSON.
- **Connexion à la base de données** : Établit une connexion à la base de données MySQL en utilisant les informations de configuration.
- **Exécution de la requête SQL** : Exécute une requête SQL pour récupérer les informations des châssis.
- **Traitement des résultats** : Traite les résultats de la requête et les retourne sous forme de dictionnaire.
- **Fermeture de la connexion** : Ferme la connexion à la base de données.

2.2 Détail des commandes

Chaque commande et fonction du script est expliquée en détail ci-dessous.

```
# -*- coding: utf-8 -*-

from mysql.connector import connect  # Imports the connect function from mysql.connector module for database connections
import json  # Imports the json module for parsing and generating JSON data

def fetch_latest_ip_addresses():
    with open('configHermes.json', 'r') as config_file:  # Opens the file in read mode
        config = json.load(config_file)  # Loads the JSON content and converts it into a Python dictionary

    # Define a connection object
    conn = connect(
        user=config['Login_Turtle']['user'],
        password=config['Login_Turtle']['password'],
        host=config['Login_Turtle']['host'],
        database=config['Login_Turtle']['database'])

```

```
17      cursor = conn.cursor()  # Create a cursor object to
            execute SQL queries.
18
19      cursor.execute("SELECT * from chassis")
20      # fetch the db results in a dictionary
21      dico = {}
22      for row in cursor.fetchall():
23          dico[f"Chassis{row[0]}"] = {"API": row[3], "
                RASP_catch": row[1]}
24
25      # Close the database connection
26      conn.commit()
27      conn.close()
28
29      return dico
```

Listing 1 – Contenu du script Python

2.2.1 Importation des modules

Le script commence par importer les modules nécessaires. `mysql.connector` est utilisé pour les connexions à la base de données et `json` pour gérer les fichiers JSON.

```
1 from mysql.connector import connect  # Imports the connect
      function from mysql.connector module for database
      connections
2 import json  # Imports the json module for parsing and
      generating JSON data
```

2.2.2 Lecture du fichier de configuration

La fonction `fetch_latest_ip_addresses` commence par ouvrir et lire le fichier de configuration JSON.

```
1 def fetch_latest_ip_addresses():
2     with open('configHermes.json', 'r') as config_file:  #
          Opens the file in read mode
3         config = json.load(config_file)  # Loads the JSON
              content and converts it into a Python dictionary
```

2.2.3 Connexion à la base de données

Ensuite, la fonction établit une connexion à la base de données MySQL en utilisant les informations de connexion extraites du fichier de configuration.

```
1     # Define a connection object
2     conn = connect(
3         user=config['Login_Turtle']['user'],
4         password=config['Login_Turtle']['password'],
5         host=config['Login_Turtle']['host'],
```

```
6         database=config['Login_Turtle']['database'])
```

2.2.4 Création et exécution de la requête SQL

La fonction crée un curseur pour exécuter des requêtes SQL et exécute une requête pour sélectionner toutes les entrées de la table `chassis`.

```
1     cursor = conn.cursor()  # Create a cursor object to
         execute SQL queries.
2
3     cursor.execute("SELECT * from chassis")
```

2.2.5 Traitement des résultats

Les résultats de la requête SQL sont ensuite traités et stockés dans un dictionnaire Python.

```
1     # fetch the db results in a dictionary
2     dico = {}
3     for row in cursor.fetchall():
4         dico[f"Chassis{row[0]}"] = {"API": row[3], "
             RASP_catch": row[1]}
```

2.2.6 Fermeture de la connexion

Enfin, la connexion à la base de données est fermée et le dictionnaire résultant est retourné.

```
1     # Close the database connection
2     conn.commit()
3     conn.close()
4
5     return dico
```

3 Conclusion

Ce script Python offre une méthode efficace pour extraire et traiter des adresses IP de châssis à partir d'une base de données MySQL, en utilisant des configurations définies dans un fichier JSON. La modularité et la simplicité de ce script en font un outil utile pour des applications similaires nécessitant des connexions à des bases de données et la manipulation de données JSON.

Références

[1] MySQL Connector/Python Documentation, `https://dev.mysql.com/doc/connector-python/en/`.

[2] Python json Module Documentation, `https://docs.python.org/3/library/json.html`.