Step 3: Proof of concept connecting to SQL using pyodbc

Article • 11/01/2023

This sample proof of concept uses `pyodbc` to connect to an SQL database. This sample assumes that you're using the AdventureWorksLT sample database .

> **Note**
>
> This example should be considered a proof of concept only. The sample code is simplified for clarity, and does not necessarily represent best practices recommended by Microsoft.

Prerequisites

- Python 3
  - If you don't already have Python, install the **Python runtime** and **Python Package Index (PyPI) package manager** from python.org .
  - Prefer to not use your own environment? Open as a devcontainer using GitHub Codespaces .
  - Open in GitHub Codespaces .
- `pyodbc` package from PyPI.
- Install the Microsoft ODBC Driver 18 for SQL Server
- An SQL database and credentials.

Connect and query data

Connect to a database using your credentials.

1. Create a new file named **app.py**.

2. Add a module docstring.

```
Python
```

```
"""
Connects to a SQL database using pyodbc
"""
```

3. Import the `pyodbc` package.

Python

```python
import pyodbc
```

4. Create variables for your connection credentials.

Python

```python
SERVER = '<server-address>'
DATABASE = '<database-name>'
USERNAME = '<username>'
PASSWORD = '<password>'
```

5. Create a connection string variable using string interpolation.

Python

```python
connectionString = f'DRIVER={{ODBC Driver 18 for SQL
Server}};SERVER={SERVER};DATABASE={DATABASE};UID={USERNAME};PWD=
{PASSWORD}'
```

6. Use the pyodbc.connect function to connect to an SQL database.

Python

```python
conn = pyodbc.connect(connectionString)
```

Execute a query

Use an SQL query string to execute a query and parse the results.

1. Create a variable for the SQL query string.

Python

```
SQL_QUERY = """
SELECT
TOP 5 c.CustomerID,
c.CompanyName,
COUNT(soh.SalesOrderID) AS OrderCount
FROM
SalesLT.Customer AS c
LEFT OUTER JOIN SalesLT.SalesOrderHeader AS soh ON c.CustomerID =
soh.CustomerID
GROUP BY
c.CustomerID,
c.CompanyName
ORDER BY
OrderCount DESC;
"""
```

2. Use cursor.execute to retrieve a result set from a query against the database.

Python

```
cursor = conn.cursor()
cursor.execute(SQL_QUERY)
```

> **Note**
>
> This function essentially accepts any query and returns a result set, which can be iterated over with the use of **cursor.fetchone()** .

3. Use cursor.fetchall with a `foreach` loop to get all the records from the database. Then print the records.

Python

```
records = cursor.fetchall()
for r in records:
    print(f"{r.CustomerID}\t{r.OrderCount}\t{r.CompanyName}")
```

4. **Save** the **app.py** file.

5. Open a terminal and test the application.

Bash

```
python app.py
```

Output

```
29485   1       Professional Sales and Service
29531   1       Remarkable Bike Store
29546   1       Bulk Discount Store
29568   1       Coalition Bike Company
29584   1       Futuristic Bikes
```

Insert a row as a transaction

In this example, you execute an INSERT statement safely and pass parameters. Passing parameters as values protects your application from SQL injection attacks.

1. Import `randrange` from the random library.

Python

```python
from random import randrange
```

2. Generate a random product number.

Python

```python
productNumber = randrange(1000)
```

> **Tip**
>
> Generating a random product number here ensures that you can run this sample multiple times.

3. Create a SQL statement string.

Python

```python
SQL_STATEMENT = """
INSERT SalesLT.Product (
Name,
```

```
ProductNumber,
StandardCost,
ListPrice,
SellStartDate
) OUTPUT INSERTED.ProductID
VALUES (?, ?, ?, ?, CURRENT_TIMESTAMP)
"""
```

4. Execute the statement using `cursor.execute`.

Python

```
cursor.execute(
    SQL_STATEMENT,
    f'Example Product {productNumber}',
    f'EXAMPLE-{productNumber}',
    100,
    200
)
```

5. Fetch the first column of the single result using cursor.fetchval, print the result's unique identifier, and then commit the operation as a transaction using connection.commit.

Python

```
resultId = cursor.fetchval()
print(f"Inserted Product ID : {resultId}")
conn.commit()
```

> **Tip**
>
> Optionally, you can use **connection.rollback** to rollback the transaction.

6. Close the cursor and connection using cursor.close and connection.close.

Python

```
cursor.close()
conn.close()
```

7. **Save** the **app.py** file and test the application again

Bash

```
python app.py
```

Output

```
Inserted Product ID : 1001
```

Next steps

Python Developer Center.