Database Programming in Python

Database Programming in Python

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- MySQL
- SQLite
- MS SQL
- PostgreSQL
- Informix
- Sybase
- Inter-base
- Oracle etc….
- The DB-API is the standard for Python's database interface.

The database is a collection of organized information that can easily be used, managed, update.

They are classified according to their organizational approach.

The Python Database interfaces are categorized into two. These are:

Generic database interface:

•Most Python's database interface remains to Python's DB-API standard

•Most of the databases have ODBC support. Other than that, the Java database usually supports JDBC, and programmers can work with that from Jython.

What Database API includes

Using Python structure, DB-API provides standard and support for working with databases.

The API consists of:

•Bring in the API module
•Obtain database connection
•Issue SQL statements and then store procedures
•Close the connection

My SQL Database

- It is an interface for associating the SQL database server from Python and uses the DB-API of Python to work.
- IMPLEMENTATION:
- To access the MySQL database using Python, you must install it on your machine; Then, type the script below to implement MySQL within your program:

```
import MySQLdb
```

```
# importing the module import MySQLdb

 # opening a database connection db = MySQLdb.connect
("localhost","testprog","stud","PYDB")

# define a cursor object cursor = conn.cursor

# drop table if exists Cursor.execute("IF STUDENT TABLE EXISTS
DROP IT")

 # query sql = "CREATE TABLE STUDENT (NAME CHAR(30)
NOT NULL, CLASS CHAR(5), AGE INT, GENDER CHAR(8),
MARKS INT"

# execute query cursor.execute(sql)

# close object cursor.close()

 # close connection conn.close()
```

Benefits of python Database Programming

•Programming in Python is considerably simple and efficient with compared to other languages, so as the database programming

•Python database is portable, and the program is also portable so both can give an advantage in case of portability

•Python supports SQL cursors

•It also supports Relational Database systems

•The API of Python for the database is compatible with other databases also

•It is platform-independent

Database operations

There are various operations that programmers can perform within the Python program. To deal with these statements, you must have a good knowledge of Database programming and SQL.

INSERT-It is an SQL statement used to create a record into a table.

READ-Fetches useful information from the database.

UPDATE-It is used to update those available or already existing record(s).

DELETE-It is used to delete records from the database.

ROLLBACK-It works like "undo", which reverts all the changes that you have made.

Disconnecting a Database

To disconnect Database connection, use close() method.

db.close()

If the connection to a database is closed by the user with the close() method, any outstanding transactions are rolled back by the DB. However, instead of depending on any of DB lower level implementation details, your application would be better off calling commit or rollback explicitly.

Handling Errors

There are many sources of errors. A few examples are a syntax error in an executed SQL statement, a connection failure, or calling the fetch method for an already canceled or finished statement handle.

The DB API defines a number of errors that must exist in each database module. The following table lists these exceptions.

Exception and Description

Warning

Used for non-fatal issues. Must subclass StandardError.

Error

Base class for errors. Must subclass StandardError.

InterfaceError

Used for errors in the database module, not the database itself. Must subclass Error.

DatabaseError

Used for errors in the database. Must subclass Error.

DataError

Subclass of DatabaseError that refers to errors in the data.

OperationalError

Subclass of DatabaseError that refers to errors such as the loss of a connection to the database. These errors are generally outside of the control of the Python scripter.

IntegrityError

Subclass of DatabaseError for situations that would damage the relational integrity, such as uniqueness constraints or foreign keys.

InternalError

Subclass of DatabaseError that refers to errors internal to the database module, such as a cursor no longer being active.

ProgrammingError

Subclass of DatabaseError that refers to errors such as a bad table name and other things that can safely be blamed on you.

NotSupportedError

Subclass of DatabaseError that refers to trying to call unsupported functionality.