

LU06.A03 - ToDo-Liste mit SQLite und DAO-Klassen

In dieser Übung wirst du eine einfache ToDo-Liste implementieren, die SQLite als Datenbank verwendet. Du wirst eine DAO-Klasse erstellen, die die CRUD-Operationen (Create, Read, Update, Delete) für die ToDo-Elemente handhabt.

Detaillierte Aufgabenstellung

- Erstelle eine `TodoItem`-Klasse mit den Attributen `item_id`, `title` und `is_completed`.
- Verwende den `@dataclass`-Dekorator für die `TodoItem`-Klasse.
- Erstelle eine `TodoDao`-Klasse, die die CRUD-Operationen für die `TodoItem`-Elemente handhabt.
- Implementiere die folgenden Funktionen in der `TodoDao`-Klasse:
 1. `create_table()`: Erstellt die Tabelle, falls sie nicht existiert.
 2. `add_item()`: Fügt ein neues ToDo-Element zur Datenbank hinzu.
 3. `get_item()`: Ruft ein ToDo-Element anhand seiner ID ab.
 4. `get_all_items()`: Ruft alle ToDo-Elemente aus der Datenbank ab.
 5. `update_item()`: Aktualisiert ein bestehendes ToDo-Element.
 6. `delete_item()`: Löscht ein ToDo-Element anhand seiner ID.
 7. `close()`: Schließt die Datenbankverbindung.

Code-Vorlage

Main

`main.py`

```
"""ToDo-Liste mit SQLite und DAO-Klassen.

Aufgabenstellung:
https://wiki.bzz.ch/modul/m323/learningunits/lu06/aufgaben/dao
"""

from todo_item import TodoItem
from todo_dao import TodoDao

def main():
    # DAO-Instanz erstellen und Tabelle erstellen
    dao = TodoDao("todo_example.db")
    dao.create_table()
```

```
# Neues ToDo-Element hinzufügen
new_item = TodoItem(None, "Buy milk", False)
dao.add_item(new_item)
new_item = TodoItem(None, "Buy cheese", False)
dao.add_item(new_item)

# ToDo-Element abrufen
retrieved_item = dao.get_item(1)
if retrieved_item:
    print(
        f"Item ID: {retrieved_item.item_id}, Title:
{retrieved_item.title}, "
        f"Is Completed: {retrieved_item.is_completed}"
    )

# Alle ToDo-Elemente abrufen
all_items = dao.get_all_items()
print("All items:")
for item in all_items:
    print(
        f"Item ID: {item.item_id}, Title: {item.title}, Is
Completed: {item.is_completed}"
    )

# ToDo-Element aktualisieren
retrieved_item.is_completed = True
is_updated = dao.update_item(retrieved_item)
print(f"Was the item updated? {is_updated}")

# ToDo-Element löschen
is_deleted = dao.delete_item(1)
print(f"Was the item deleted? {is_deleted}")

# Verbindung schließen
dao.close()

# Ausführung der Hauptlogik
if __name__ == "__main__":
    main()
```

todo_dao

todo_dao.py

```
"""Datenbankzugriff für die ToDo-Einträge."""
```

```
import sqlite3
from todo_item import TodoItem

# TODO: Implementiere die TodoDao-Klasse für CRUD-Operationen
class TodoDao:
    """Liest und schreibt ToDo-Einträge in der SQLite-Datenbank."""
```

todo_item

todo_item.py

```
"""Datenmodell eines ToDo-Eintrags."""

from dataclasses import dataclass

# TODO: Implementiere die TodoItem-Klasse mit @dataclass
@dataclass
class TodoItem:
    ...
```

Vorgehen

1. Akzeptiere das GitHub Classroom Assignment
2. Kclone dein persönliches Repository in die Entwicklungsumgebung
3. Beginne mit der Implementierung der TodoItem-Klasse. Verwende den @dataclass-Dekorator.
4. Erstelle die TodoDao-Klasse und implementiere die Methode create_table().
5. Füge die Methoden add_item(), get_item(), get_all_items(), update_item() und delete_item() zur TodoDao-Klasse hinzu.
6. Teste alle Methoden, um sicherzustellen, dass sie wie erwartet funktionieren.

Abgabe

Die Abgabe der Lösung erfolgt als Push in das persönliche GitHub-Repository.

⇒ *GitHub Repo für externe Besucher*

GitHub Repository <https://github.com/templates-python/m323-lu06-a03-dao>

Lernende am BZZ müssen den Link zum GitHub Classroom Assignment verwenden

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/modul/m323/learningunits/lu06/aufgaben/dao>

Last update: **2026/08/17 10:18**

