Schritt-für-Schritt-Anleitung: Wechsel zu einer neuen MySQL-Datenbank in Django

Diese Anleitung beschreibt, wie du in deinem Django-Projekt eine neue MySQL-Datenbank einrichtest, einen eigenen Benutzer für die Datenbank erstellst und Daten aus der alten Datenbank in die neue überträgst.

---

1. Voraussetzungen

1. **MySQL ist installiert** und läuft auf deinem System.
2. **Python-Umgebung ist eingerichtet**, und Django ist installiert.
3. Du hast Zugriff auf die bestehende SQLite-Datenbank.

---

2. Neue MySQL-Datenbank und Benutzer erstellen

2.1 MySQL-Client öffnen

Melde dich mit Administratorrechten in MySQL an:

```
mysql -u root -p
```

2.2 Neue Datenbank erstellen

Erstelle eine neue Datenbank für dein Projekt, z. B. `movies`:

```
CREATE DATABASE movies CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

2.3 Benutzer für die neue Datenbank erstellen

Erstelle einen neuen Benutzer, z. B. `movies_user`:

```
CREATE USER 'movies_user'@'localhost' IDENTIFIED BY 'dein_passwort';
```

2.4 Berechtigungen zuweisen

Gewähre dem Benutzer alle Rechte für die neue Datenbank:

```
GRANT ALL PRIVILEGES ON movies.* TO 'movies_user'@'localhost';
FLUSH PRIVILEGES;
```

---

3. Django für MySQL konfigurieren

3.1 `mysqlclient` installieren

Installiere die MySQL-Datenbank-Bibliothek für Django:

```
pip install mysqlclient
```

3.2 `settings.py` anpassen

Bearbeite die Datenbankeinstellungen in der `settings.py` deines Django-Projekts:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'movies',  # Name der neuen Datenbank
        'USER': 'movies_user',  # Benutzername
        'PASSWORD': 'dein_passwort',  # Passwort
        'HOST': 'localhost',  # Standard-Host
        'PORT': '3306',  # Standard-Port
    }
}
```

3.3 Migrationen durchführen

Führe die Migrationen aus, um die Tabellenstruktur in der neuen Datenbank zu erstellen:

```
python manage.py makemigrations
python manage.py migrate
```

---

4. Daten von SQLite nach MySQL übertragen

4.1 Daten aus SQLite exportieren

Exportiere die Daten aus der alten SQLite-Datenbank in eine JSON-Datei:

```
python manage.py dumpdata > data.json
```

4.2 Daten in MySQL importieren

Importiere die Daten in die neue MySQL-Datenbank:

```
python manage.py loaddata data.json
```

---

5. Testen

1. Starte den Django-Server, um zu prüfen, ob die neue Datenbank funktioniert:

   ```
   python manage.py runserver
   ```

2. Überprüfe die Daten im Django-Admin-Panel.

---

6. Zusätzliche Hinweise

Projekte mit separaten Datenbanken

Jedes Django-Projekt kann eine eigene Datenbank und einen eigenen Benutzer haben. Wiederhole die Schritte 2-4, um für jedes Projekt eine eigene Datenbank mit Benutzer zu erstellen.

Sicherheit

- Speichere Passwörter nicht direkt in der `settings.py`. Verwende stattdessen Umgebungsvariablen oder ein `.env`-File.

---

7. Fehlerbehebung

- **Fehler bei der Verbindung zur MySQL-Datenbank:** Überprüfe die MySQL-Dienstkonfiguration:

  ```
  sudo service mysql start
  ```

- **Fehler beim Importieren von Daten:** Stelle sicher, dass alle Apps aktiviert sind und dass die JSON-Daten die richtige Struktur haben.

---

8. Beispiel für mehrere Projekte

Projekt 1: movies

- Datenbank: `movies`
- Benutzer: `movies_user`

Projekt 2: blog

- Datenbank: `blog`
- Benutzer: `blog_user`

Führe für jedes Projekt die gleichen Schritte aus und passe die Namen entsprechend an.

Mit dieser Anleitung kannst du erfolgreich von SQLite auf MySQL wechseln und gleichzeitig für jedes Django-Projekt separate Datenbanken und Benutzer einrichten.