Python und Datenbanken

Datenbanken aus Python zuverlässig und strukturiert nutzen

01 INTRO

Viele Python-Anwendungen müssen Daten dauerhaft speichern, abfragen und verändern. Dafür reicht es nicht, SQL aus einem Python-Programm heraus aufzurufen: Verbindungen, Transaktionen, Parameter, Fehlerbehandlung und die Struktur der Datenzugriffsschicht müssen ebenfalls berücksichtigt werden.

Der Kurs zeigt den praktischen Datenbankzugriff mit Python – vom direkten Arbeiten mit SQL bis zum Einsatz eines ORM. Im Mittelpunkt stehen relationale Datenbanken, saubere Schnittstellen zwischen Anwendung und Persistenz sowie robuster und nachvollziehbarer Datenbankcode.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Sichere Python-Grundkenntnisse, grundlegende SQL-Kenntnisse

02 TRAININGSPROGRAMM

Inhalte

Python und relationale Datenbanken

- Aufgaben einer Datenbank in Anwendungen
- Python DB-API
- Datenbanktreiber
- Verbindungen aufbauen und schließen
- Cursor
- SQL aus Python ausführen
- Abfrageergebnisse verarbeiten
- Datenbankzugriffe sinnvoll kapseln

Daten lesen und verändern

- SELECT, INSERT, UPDATE und DELETE
- einzelne und mehrere Datensätze verarbeiten
- Parameter an SQL-Anweisungen übergeben
- Rückgabewerte und erzeugte IDs
- NULL und None
- Datentypen zwischen Python und Datenbank
- Batch-Verarbeitung

Sichere SQL-Ausführung

- SQL und Daten voneinander trennen
- parametrisierte Statements
- SQL Injection
- dynamische Abfragen
- Benutzereingaben sicher verarbeiten
- Identifier und Werte unterscheiden

Transaktionen

- Grundlagen von Transaktionen
- Commit und Rollback
- atomare Änderungen
- Fehler innerhalb einer Transaktion
- Transaktionsgrenzen festlegen
- Context Manager für Datenbankzugriffe
- typische Transaktionsfehler

Verbindungen und Ressourcen

- Lebenszyklus einer Datenbankverbindung
- Verbindungen zuverlässig schließen
- Connection Pooling
- Timeouts
- unterbrochene Verbindungen
- parallele Datenbankzugriffe
- Ressourcenverbrauch berücksichtigen

Datenzugriff strukturieren

- SQL nicht über die Anwendung verteilen
- Funktionen und Klassen für Datenzugriffe
- Repository-Ansatz
- Fachlogik und Persistenz trennen
- Datenbankmodelle und Domänenobjekte
- Abhängigkeiten kontrollieren
- testbare Datenzugriffsschichten

Einstieg in SQLAlchemy

- Aufgaben von SQLAlchemy
- Engine und Connection
- SQLAlchemy Core
- SQL-Ausdrücke
- Tabellen und Metadaten
- Abfragen ausführen
- Ergebnisse verarbeiten
- Transaktionen mit SQLAlchemy

Object Relational Mapping

- ORM-Grundidee
- deklarative Modelle
- Tabellen auf Python-Klassen abbilden
- Sessions
- Objekte speichern, laden und ändern
- Beziehungen zwischen Objekten
- One-to-Many und Many-to-Many
- Cascades im Überblick

Abfragen mit dem ORM

- Objekte filtern und sortieren
- Joins
- Beziehungen laden
- Lazy und Eager Loading
- Aggregationen
- Pagination
- typische ORM-Fallen
- N+1-Queries erkennen

Datenbankschema und Migrationen

- Anwendung und Datenbankschema
- Schemaänderungen
- Migrationen
- Alembic im Überblick
- Migrationen erzeugen und ausführen
- Änderungen nachvollziehbar halten
- Migrationen in unterschiedlichen Umgebungen

Fehlerbehandlung und Diagnose

- Datenbankfehler behandeln
- Constraint-Verletzungen
- Verbindungsfehler
- Rollbacks nach Fehlern
- Logging von SQL
- langsame Abfragen erkennen
- Datenbankprobleme von Anwendungsfehlern unterscheiden

Datenbankzugriffe testen

- Datenzugriffsschichten testbar gestalten
- Testdatenbanken
- Testdaten erzeugen
- Tests voneinander isolieren
- Transaktionen in Tests
- Integrationstests
- Mocking von Datenbankzugriffen sinnvoll einsetzen

Auswahl des geeigneten Ansatzes

- direktes SQL
- SQLAlchemy Core
- ORM
- Vor- und Nachteile der Ansätze
- Performance und Wartbarkeit
- bestehende Datenbankschemata
- unterschiedliche Anforderungen verschiedener Anwendungen

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit aktueller Python-Version und Entwicklungsumgebung

> **Inhouse-Schulungen**
> Die Schwerpunkte können an vorhandene Kenntnisse, eingesetzte Technologien und konkrete Anforderungen des Teams angepasst werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Python-Entwicklerinnen und -Entwickler, die Anwendungen mit relationalen Datenbanken verbinden und Datenbankzugriffe strukturiert implementieren möchten.

Er eignet sich sowohl für Teilnehmer, die bisher hauptsächlich direkt mit SQL arbeiten, als auch für Entwickler, die SQLAlchemy oder ein ORM gezielter einsetzen und die zugrunde liegenden Mechanismen besser verstehen möchten.

Die Teilnehmer sollten mit folgenden Themen sicher umgehen können:

- grundlegende Python-Syntax
- Funktionen
- Module und Packages
- Klassen und Objekte
- Exceptions

- Context Manager in Grundzügen

> **SQL-Kenntnisse**
> Grundlegende SQL-Kenntnisse wie SELECT, INSERT, UPDATE und DELETE werden vorausgesetzt.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- relationale Datenbanken aus Python ansprechen
- SQL-Anweisungen sicher parametrisieren
- Abfrageergebnisse und Datenbanktypen korrekt verarbeiten
- Transaktionen bewusst einsetzen
- Verbindungen und Datenbankressourcen zuverlässig verwalten
- Datenbankzugriffe von der Fachlogik trennen
- SQLAlchemy Core und ORM für unterschiedliche Anforderungen einsetzen
- Beziehungen zwischen Datenbankobjekten modellieren
- typische ORM-Probleme wie unnötige oder zu viele Abfragen erkennen
- Schemaänderungen mit Migrationen verwalten
- Datenbankfehler systematisch behandeln und diagnostizieren
- Datenbankzugriffe automatisiert testen
- zwischen direktem SQL, SQLAlchemy Core und ORM situationsgerecht entscheiden

06 FAQ

Häufige Fragen

Welche Datenbank wird im Kurs verwendet?

Die grundlegenden Konzepte sind weitgehend unabhängig vom Datenbanksystem. Für die Übungen kann beispielsweise PostgreSQL oder eine andere relationale Datenbank eingesetzt werden. Bei Inhouse-Schulungen kann die vorhandene Datenbank berücksichtigt werden.

Muss ich SQL bereits beherrschen?

Grundlegende SQL-Kenntnisse werden vorausgesetzt. Der Kurs ist keine SQL-Grundlagenschulung, behandelt SQL aber dort, wo es für den Datenbankzugriff aus Python erforderlich ist.

Wird SQLAlchemy behandelt?

Ja. Sowohl SQLAlchemy Core als auch das ORM werden behandelt. Dadurch können die Teilnehmer die unterschiedlichen Ansätze vergleichen und situationsgerecht einsetzen.

Geht es ausschließlich um ORMs?

Nein. Direkter SQL-Zugriff ist ausdrücklich Bestandteil des Kurses. Ein Ziel ist gerade, nicht jede Datenbankaufgabe automatisch mit einem ORM zu lösen.

Werden auch NoSQL-Datenbanken behandelt?

Nein. Der Schwerpunkt liegt auf relationalen Datenbanken. Dokumenten-, Key-Value- oder andere NoSQL-Datenbanken erfordern teilweise andere Konzepte und sind nicht Gegenstand dieses Kurses.

Werden Datenbankmigrationen behandelt?

Ja. Schemaänderungen und Migrationen werden behandelt, einschließlich eines Einstiegs in Alembic.

Kann der Kurs mit unserer vorhandenen Datenbank durchgeführt werden?

Ja. Bei Inhouse-Schulungen können Datenbanksystem, Treiber, bestehende Schemata und typische Zugriffsanforderungen des Unternehmens berücksichtigt werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/python-datenbanken/