SQL – Java

- JDBC
- DAO
- Esempio:
  - https://github.com/egalli64/mjd
    - Maven
    - MySQL 8

JDBC

- Permette di connettere un progetto Java a un database
- Si aggiunge la dipendenza JDBC per il database scelto
  - MySQL: Connector Java per la versione corrente
    - Ad es: mysql-connector-java 8.0.x
  - Oracle: ojdbc8 certificato per JDK 8, ojdbc11 …
- Si scrive codice Java usando le interfacce definite nei package

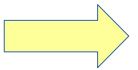
  java.sql

  javax.sql

DriverManager e Connection

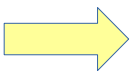
- Servizio di base che gestisce i database driver presenti nel progetto
- getConnection()
  - url, segue le specifiche fornite dal DBMS utilizzato
    - jdbc:oracle:thin:@127.0.0.1:1521/xe
    - jdbc:mysql://localhost:3306/me?serverTimezone=Europe/Rome
- Connection
  - Estende l'interfaccia AutoCloseable
  - Media lo scambio di dati tra Java e database

```
Connection conn = DriverManager.getConnection(url, user, password);
```



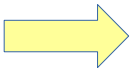
Statement

- Rappresenta un comando da eseguire sul database
  - execute() per DDL, true se genera un ResultSet associato
  - executeUpdate() per DML, ritorna il numero di righe interessate
  - executeQuery() per SELECT, ritorna il ResultSet relativo
- Generato da un oggetto Connection per mezzo del metodo createStatement()
- Estende l'interfaccia AutoCloseable
- Lo Statement implica tipicamente quattro passi per il suo svolgimento:
  - Parse, compilazione, pianificazione e ottimizzazione, esecuzione della query



PreparedStatement

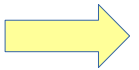
- Estensione di Statement
- Connection.prepareStatement()
  - Uso di ‘?’ come segnaposto per i parametri
- Esecuzione anticipata di parse, compilazione e ottimizzazione della query
- Lo statement può essere precompilato e salvato in cache dal DBMS
- Protezione da attacchi via SQL injection usando i suoi setter per i parametri
- Rende più semplice la conversione di tipi non standard tra Java e SQL



ResultSet

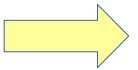
- Una tabella di dati che rappresenta il result set ritornato dal database
- Estende l'interfaccia AutoCloseable
- Per default, non supporta update e può essere percorso solo in modalità forward
- Normalmente ottenuto da uno Statement via executeQuery()

```
ResultSet rs = stmt.executeQuery("SELECT coder_id, first_name, last_name FROM coders");
```



SQLException

- Rappresenta un errore generato da JDBC
- Qualcosa non ha funzionato nell’accesso a database, o altri problemi
- Possiamo assumere che tutto il nostro codice JDBC richieda di essere eseguito in blocchi try/ catch per questa eccezione



SELECT via JDBC

try with resources

```
try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
        Statement stmt = conn.createStatement()) {
    ResultSet rs = stmt.executeQuery("SELECT first_name FROM coders ORDER BY 1");

    List<String> results = new ArrayList<String>();
    while (rs.next()) {
        results.add(rs.getString(1));
    }

    // …
}
```

executeQuery() on SELECT

itera sul result set

legge la prima colonna della riga corrente del result set come stringa

SELECT via JDBC (prepared)

```
/* String CODERS_BY_SALARY = "SELECT first_name, last_name, salary FROM coders "
            + "WHERE salary >= ? ORDER BY 3 DESC"; */

// …

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD); //
        PreparedStatement prepStmt = conn.prepareStatement(CODERS_BY_SALARY)) {
    prepStmt.setDouble(1, lower);

    ResultSet rs = prepStmt.executeQuery();

    List<Coder> results = new ArrayList<>();
    while (rs.next()) {
        results.add(new Coder(rs.getString(1), rs.getString(2), rs.getInt(3)));
    }

    // …
}
```

Transazioni

- By default, una connessione è in modalità autocommit, ogni statement viene committato
- Connection.setAutoCommit(boolean)
- Connection.commit()
- Connection.rollback()

Oggetti – Relazioni

- Java è un linguaggio Object-Oriented
  - Gestione della logica dell'applicazione, gestisce i dati usando
    - Ereditarietà
    - Accesso a dati correlati via collezioni o relazioni *has-a*
    - Tipi di dato propri del linguaggio (int, String, …)
- MySQL è un DBMS relazionale
  - Modella i dati usati dall'applicazione, gestisce i dati usando
    - Tipi di dato SQL (VARCHAR, DECIMAL, …)
    - JOIN, Primary Key, Foreign Key

Operazioni CRUD

- Create: inserimento di un oggetto/record nel database via INSERT
- Read: lettura di un record/oggetto via SELECT
  - Per chiave
  - Per attributo
    - Potrebbe ritornare una collezione di oggetti
- Update: aggiornamento di un record/oggetto via UPDATE
- Delete: eliminazione di un record via DELETE

Data Access Object (DAO)

- Pattern per semplificare l’accesso a data source
  - DAL: Data Access Layer
- Problema
  - L’implementazione delle funzionalità CRUD (Create Read Update Delete) dipende dal DBMS utilizzato
- Soluzione
  - La logica di accesso ai dati è isolata nell’oggetto DAO
- Conseguenze
  - Miglior portabilità dell’applicazione per diversi DBMS
  - Il codice all’esterno del DAO è indipendente dal data souce

DTO e DAO

- Data Transfer Object
  - JavaBean
  - Valori scambiati tra database e applicazione
- Data Access Object
  - Interfaccia che dichiara le operazioni disponibili (CRUD et al.)
- Oggetto DAO
  - Implementa l’interfaccia per un particolare database
  - Tipicamente istanziato via factory method