

Esercitazione 6 – JDBC

- Usare gli strumenti di Mysql per

1) creare un database con nome MapDB

```
CREATE DATABASE MapDB;
```

2) creare l'utente "MapUser" con password "map" che sia dotato dei diritti di accesso in lettura e scrittura per mapDB

```
CREATE USER 'MapUser'@'localhost' IDENTIFIED BY 'map';
```

```
GRANT CREATE, SELECT, INSERT, DELETE ON MapDB.* TO MapUser@localhost IDENTIFIED BY 'map';
```

2) creare e popolare la tabella denominata playtennis

```
CREATE TABLE MapDB.playtennis(  
    outlook varchar(10),  
    temperature float(5,2),  
    umidity varchar(10),  
    wind varchar(10),  
    play varchar(10)  
);  
  
insert into MapDB.playtennis values('sunny',30.3,'high','weak','no');  
insert into MapDB.playtennis values('sunny',30.3,'high','strong','no');  
insert into MapDB.playtennis values('overcast',30.0,'high','weak','yes');  
insert into MapDB.playtennis values('rain',13.0,'high','weak','yes');  
insert into MapDB.playtennis values('rain',0.0,'normal','weak','yes');  
insert into MapDB.playtennis values('rain',0.0,'normal','strong','no');  
insert into MapDB.playtennis values('overcast',0.1,'normal','strong','yes');  
insert into MapDB.playtennis values('sunny',13.0,'high','weak','no');  
insert into MapDB.playtennis values('sunny',0.1,'normal','weak','yes');  
insert into MapDB.playtennis values('rain',12.0,'normal','weak','yes');  
insert into MapDB.playtennis values('sunny',12.5,'normal','strong','yes');  
insert into MapDB.playtennis values('overcast',12.5,'high','strong','yes');  
insert into MapDB.playtennis values('overcast',29.21,'normal','weak','yes');  
insert into MapDB.playtennis values('rain',12.5,'high','strong','no');
```

- Nel progetto QT creare il package `database` da popolare con le classi: `DbAccess`, `TableData`, `TableSchema`, `DatabaseConnectionException`, `NoValueException`, `QUERY_TYPE`, `Example`, `EmptyTypeException`

- Definire la classe `DatabaseConnectionException` che estende `Exception` per modellare il fallimento nella connessione al database.
- Definire la classe `DbAccess` che realizza l'accesso alla base di dati.

Attributi

private String DRIVER_CLASS_NAME = "org.gjt.mm.mysql.Driver"; (Per utilizzare questo Driver scaricare e aggiungere al classpath il connettore mysql connector)

private final String DBMS = "jdbc:mysql";

private final String SERVER="localhost"; contiene l'identificativo del server su cui risiede la base di dati (per esempio localhost)

private final String DATABASE = "MapDB"; contiene il nome della base di dati

private final String PORT=3306; La porta su cui il DBMS MySQL accetta le connessioni

private final String USER_ID = "MapUser"; contiene il nome dell'utente per l'accesso alla base di dati

private final String PASSWORD = "map"; contiene la password di autenticazione per l'utente identificato da USER_ID

private Connection conn; gestisce una connessione

Metodi

public void initConnection() throws DatabaseConnectionException; impartisce al class loader l'ordine di caricare il driver mysql, inizializza la connessione riferita da *conn*. Il metodo solleva e propaga una eccezione di tipo `DatabaseConnectionException` in caso di fallimento nella connessione al database. Suggestione: La stringa di connessione sarà: *DBMS+"://" + SERVER + ":" + PORT + "/" + DATABASE*

public Connection getConnection(); restituisce *conn*;

public void closeConnection(); chiude la connessione *conn*;

- Definire la classe `Table_Schema` (fornita da docente) che modella lo schema di una tabella nel database relazionale
- Definire la classe `NoValueException` che estende `Exception` per modellare l'assenza di un valore all'interno di un resultset

- Definire la classe `EmptySetException` che estende `Exception` per modellare la restituzione di un resultset vuoto.
- Definire la classe `Example` (fornita dal docente) che modella una transazione letta dalla base di dati.
- Definire la classe `Table_Data` (parzialmente fornita dal docente) che modella l'insieme di transazioni collezionate in una tabella. La singola transazione è modellata dalla classe `Example`.

```
public List<Example> getDistinctTransazioni(String table) throws SQLException, EmptySetException
```

Input: nome della tabella nel database.

Output: Lista di transazioni distinte memorizzate nella tabella.

Comportamento: : Ricava lo schema della tabella con nome `table`. Esegue una interrogazione per estrarre le tuple *distinte* da tale tabella. Per ogni tupla del resultset, si crea un oggetto, istanza della classe `Example`, il cui riferimento va incluso nella lista da restituire. In particolare, per la tupla corrente nel resultset, si estraggono i valori dei singoli campi (usando `getFloat()` o `getString()`), e li si aggiungono all'oggetto istanza della classe `Example` che si sta costruendo.

Il metodo può propagare un'eccezione di tipo `SQLException` (in presenza di errori nella esecuzione della query) o `EmptySetException` (se il resultset è vuoto)

```
public Set<Object> getDistinctColumnValues (String table, Column column) throws SQLException
```

Input: Nome della tabella, nome della colonna nella tabella

Output: Insieme di valori distinti ordinati in modalità ascendente che l'attributo identificato da nome `column` assume nella tabella identificata dal nome `table`.

Comportamento: Formula ed esegue una interrogazione SQL per estrarre i valori distinti ordinati di `column` e popolare un insieme da restituire (*scegliere opportunamente in Set da utilizzare*).

```
public Object getAggregateColumnValue(String table, Column column, QUERY_TYPE aggregate) throws SQLException, NoValueException
```

Input: Nome di tabella, nome di colonna , operatore SQL di aggregazione (min,max)

Output: Aggregato cercato.

Comportamento: Formula ed esegue una interrogazione SQL per estrarre il valore aggregato (valore minimo o valore massimo) cercato nella colonna di nome `column` della tabella di nome `table`. Il metodo solleva e propaga una `NoValueException` se il resultset è vuoto o il valore calcolato è pari a null

N.B. `aggregate` è di tipo `QUERY_TYPE` dove `QUERY_TYPE` è la classe enumerativa fornita

dal docente

```
public enum QUERY_TYPE {  
    MIN, MAX  
}
```

- Modificare nella classe **Data** la dichiarazione del membro **data** come segue

```
private List<Example> data=new ArrayList<Example>();
```

- Rimpiazzare il costruttore della classe **Data** con un costruttore che si occupa di caricare i dati di addestramento da una tabella della base di dati. Il nome della tabella è un parametro del costruttore.
- Modificare **MainTest** in modo da acquisire il nome della tabella contenente i dati per l'apprendimento da tastiera.