# ORM and Hibernate

Bernardo Cuteri

# PERSISTENCE

Enterprise applications need **persistence**

- they require data to be persistent and not lost when the application is closed or the system is powered off
- relational databases are in a descending trend, but are still the de-facto standard and their popularity is still sitting over the 80%
- several RDBMS exist (e.g. oracle, mysql)
- SQL is the leading language of RDBMS querying and manipulation

# Persistence in Java Applications

The Java Database Connectivity (**JDBC**) API is the standard way to access a DB in Java

1. prepare query and set parameters
2. execute the query
3. retrieve values from the result set

# Data Transfer Object (DTO)

- transfers data between classes and modules of the application
- usually are java beans

# Data Access Object (DAO)

- provides an abstract interface to data operations without exposing database details
- provides CRUD operations
  - Create
  - Read
  - Update
  - Delete

# Object/Relational Mapping (ORM)

- Domain entities and relations are represented in terms of **classes** (in OO programming code) and **tables** (in relational databases)
- the two representations need to be mapped from one to the other and viceversa when the application needs persistence
- ORM is the automated mapping of objects in a Java application to tuples in the tables in a relational database
- the mapping is described with **metadata** (typically annotations in java classes or xml files)

# What is Hibernate?

- an open source ORM library for Java
- supports either XML configuration files or annotations
- the user can focus on modelling the application domain in java classes
- it is not needed to write hand written SQL code for the persistence of objects and even for the database schema creation

# Some important Hibernate classes

- **Transaction** Wraps a JDBC transaction. It represent a single operation on the Database.
- **Session** Is a dialogue between the application and the persistence layer and may contain multiple transactions.
- **SessionFactory** It handles the session creation task

Typical interaction:

1. Instantiate and open a session using a session factory
2. Execute transactions
3. Close the session