ORM

Saturday, 18 February, 2023

Module Content

1. Introduction to ORM & Hibernate
    1. Architectural overview

2. Bootstrapping
    1. Native bootstrapping

3. Domain model
    1. Mapping types
    2. Basic types
    3. Identifiers
    4. Associations

4. Scheme generation

5. Persistance context
    1. Persistance entities
    2. Deleting entities
    3. Detach state
    4. Dirty checking
    5. Refresh entity state
    6. Cascading entity state

6. Transaction management

7. Data fetching
    1. Eager fetching
    2. Lazy fetching

8. Query management
    1. HQL
    2. JPQL
    3. Native SQL

Prerequisites

A. Objects
B. Classes
C. Usage of object orientation
D. Encapsulation
E. Inheritance (Is a relationalships, Has a relationaships)

DBMS related concepts

A. Ability to perform basic SQL CRUD operations
B. Basic idea of Entity relationalship types
    1. One to one
    2. One to many
    3. Many to one
    4. Many to many
    5. N-ary relationalships
C. Ability to explain the usage of JDBC
D. Basic idea of transaction management

Introduction to ORM & Hibernate

ORM stands for Object Relational Mapping.

It is a technique used in creating a "bridge" in between OOP & relational database.

? Why ORM?

1. In previous application we developed,
    - For our programming purposes, we used OOP languages (Java) where we deal with Object Model.
    - For our data storing perspective, we used relational databases (MySQL) where we deal with relational model.

2. So, here we can see that there is a huge mismatch between these 2 models mentioned above (Object models vs Relational model) where Relational databases (RDBMS) represents data is Tabular format while OOP language represent it as an interconnected graph of objects (Object model).

3. So, that's where ORM comes in to the picture to solve the above problem which works as an intermediate tool which converts data between both of these 2 models (Object Model vs Relational Model).

? How ORM works?

1. The main postulate that characterized ORM is that it encapsulates database interaction inside an Object.

2. One part of the object keeps the data and the other part knows how to deal with the data and transfers it to the relational database (This functionality is implementer inside ORM Search Engine).

3. ORM tool internally uses the JDBC API to interact with the database.

? What is ORM tool?

- Simply an ORM tool is a software that is designed to help OOP developers to interact with relational database.
- So, instead of creating your own ORM software from scratch, you can use those tools.
- So from the below example you may see the what ORM exactly does along with the normal native query for the following scenario.
  Ex:

```
SELECT id, name, email, country, phone_number, FROM user WHERE id = 20;

userRepository.findById(20); // JPA ORM Solution
```

- So the code above does the same as the SQL query.

? What is an ORM tools?

- Kindly note that every ORM tool is built differently so the methods are never the same, but the general purpose is similar.
- Your ORM tool may also provide a similar method as mentioned above.
- ORM tool internally uses the JDBC API to interact with the database.

ORM Tools

- Most OOP languages have variety of ORM tools that you can choose from.
- Here's some popular ORM tools for JAVA.

1. Hibernate
2. Apache Open JPA
3. EclipseLink
4. JOOQ
5. Oracle TopLink

Advantages & Disadvantages using ORM tools.

Advantages
- Speed up the development time for teams (Save time).
- Decrease the cost of development.
- Handle the logic required to interact with the database.
- Improve security. ORM tools are built to eliminate the possible or SQL injection attacks.
- You have to write less code when using ORM with SQL.

Disadvantages
- Learning how to use ORM can be time consuming (Learning curve).
- ORM generally slower than using SQL (Due to inside operation).
- They are not going to perform better when very complex queries are involved.

? Introduction to Hibernate, What, When, Who, Why?

Introduction to Hibernate

Hibernate is a java framework that simplifies the development of java application to interact with the database.
It was started in 2001 by Gavin King as an alternative for EJB2 style entity bean.
It’s an opensource, Lightweight , ORM tool.
Hibernate implements the specification of JPA (Java Persistence API) for data persistance.

What is JPA?

Java Persistance API is a java specification that provides certain functionality and standards to ORM tools.

The **javax.persistane** package contains the JPA classes and interfaces.

Advantage of hibernate framework

- Open source and lightweight
- Fast performance
- Database independent query
- Automatic table generation
- Simplifies complex join
- Provides query statics and database status

JDBC vs Hibernate

JDBC is a connectivity tool. Hibernate can perform automatic object perform. It's maps the object model's data to the scheme of the database itself with the help of annotation. In JDBC, one needs to write code to map the object model's data representation to the scheme of the relational model.

1. Import packages
   Import.java.sql.Connection

Import.java.sql.DriverManager
Import.java.sql.SQLExeception

2. Register database driver to JDBC
   Class.forName("Database Driver")

3. Open Connection
   Connection connection = Driver

4. Create statement

5. Execute query

6. Extract data from ResultSet

7. Close Connection

Hibernate

1. Crate session factory
2. Create a session
3. Begin transaction from session
4. Excute query & commit transaction
5. Close connection

Architectural Overview(Hibernate)

Hibernate has a layered architecture which helps the user to operate without having to know the underlaying APIs.

Hibernate makes use of the database and configuration data to provide persistence services and persistence object to the application.

This is a height level view of Hibernate Application Architecture.

This is a detailed view of the Hibernate Application architecture with its important core class.

Hibernate uses various existing java APIs, like JDBC, Java Transaction API(JTA), and java naming and directory interface (JNDI), JDBC provides a primary level of abstraction of functionality common to relational databases with a JDBC driver Hibernate to be integrated with J2EF application server.

Class objects involved in Hibernate applied architecture

Configuration Object

The Configuration object is the first Hibernate object you create in any Hibernate application. It is usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.
The Configuration object provides two keys components –

- Database Connection – This is handled through one or more configuration files supported by Hibernate. These files are hibernate.properties and hibernate.cfg.xml.

- Class Mapping Setup – This component creates the connection between the Java classes and database tables.

SessionFactory Object

Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.
The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Session Object

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.
The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.

Transaction Object

A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).
This is an optional object and Hibernate applications may choose not to use this interface, instead managing transactions in their own application code.

Query Object

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Criteria Object

Criteria objects are used to create and execute object oriented criteria queries to retrieve objects.

From <https://www.tutorialspoint.com/hibernate/hibernate_architecture.htm>

Bootstrapping

Bootstrapping refuse to the process of building and initializing a SessionFactory.

To achieve this purpose, we need to have a ServiceRegistry that holds the Services needed by Hibernate.

From this registry. We can build a Metadata object that represents the application's domain model and mapping to the database.

According to the hibernate there are two ways to perform the configuration.

1. XML configuration
2. Property life configuration