Hibernate Overview

By

Khader Shaik

Agenda

- Introduction to ORM
- Overview of Hibernate
- Why Hibernate
- Anatomy of Example
- Overview of HQL
- Architecture Overview
- Comparison with iBatis and JPA

Introduction to ORM

- Object Persistence
  - Saving objects for future use
  - Storage could be a File system, RDBMS etc
  - Today's popular data storage systems are RDBMS
  - Objects are not directly mapped to RDBMS tables
- Traditional Solutions
  - JDBC/SQL code embedded in Class, EJB (J2EE) solution etc
  - More coding, container dependent etc are the issues
  - Best practice would be to keep the Persistence separate from classes

Introduction to ORM cont..

- ORM - Object Relational Mapping – New solution
  - Persists Objects in a Relational Database
  - Transparent solution; underlying tables are hidden from classes
  - Support CRUD (Create, Read, Update and Delete) operations
  - Provides RDBMS Vendor independence
- ORM Solutions
  - Hibernate – Open Source
  - iBatis SQL Maps – Open Source
  - TopLink – Commercial
  - JPA – Java EE 5 Solution

Overview of Hibernate

- Open Source light-weight ORM solution
- Doesn't require container (light-weight)
- Object based model
- Transparent solutions
- It is around from quite some time
- Very well matured and adopted by a large developer community
- Latest Version 3.x

Why Hibernate?

- Hibernate was introduced to address the issues of Entity Beans
- Hibernate is built on top of JNDI, JDBC, JTA
- It uses XML based configuration files for mapping
- Supports many databases like Sybase, Oracle, MySQL, other Object Oriented Databases etc.
- Easy migration from one vendor database to another
- Hibernate generates the JDBC Code based on the underlying vendor database
- Hibernate APIs are very simple to learn and use
- Provides quite powerful object query language known as Hibernate Query Language (HQL)

Example - Java Class

```
public class Trade {
        private Long tradeId;
        private String clientId;
        private String symbol;
        private String orderType;

        public Long getTradeId() { return id; }
        private void setTradeId(Long id) { this.tradeId = id; }
        public String getSymbol() { return symbol; }
        public void setSymbol(String text) { this.symbol = text; }
}
```

Hibernate Persistence Code

```
…
//Initialize Hibernate session
Session session = getSessionFactory().openSession();
//Start the transaction
Transaction tx = session.beginTransaction();
//Object to be persisted
Trade trade = new Trade();
//Set the object values
//Persist the Object
session.save(trade);
//Commit the transaction
tx.commit();
//Close the Hibernate Session
session.close();
….
```

How is Hibernate Persisting?

- Hibernate used XML Mapping file to generate the SQL code to save the object

```
<hibernate-mapping>
        <class name="Trade" table="Trades">
                <id name="tradeId" column="TRADE_ID"></id>
                <property name="clientId" column="CLIENT_ID"/>
                <property name="symbol" column="SYMBOL"/>
                <property name="orderType" column="ORDER_TYPE"/>
        </class>
</hibernate-mapping>
```

Hibernate Generates SQL Statement like

```
INSERT INTO Trades (TRADE_ID, CLIENT_ID, SYMBOL, ORDER_TYPE)
VALUES (30, "CL7678", "IBM", "M")
```

RDBMS Table

```
Create Table Trades(
    TRADE_ID int not null,
    CLIENT_ID varchar(50),
    SYMBOL varchar(15),
    ORDER_TYPE char(1)
    )
```

HQL - Example

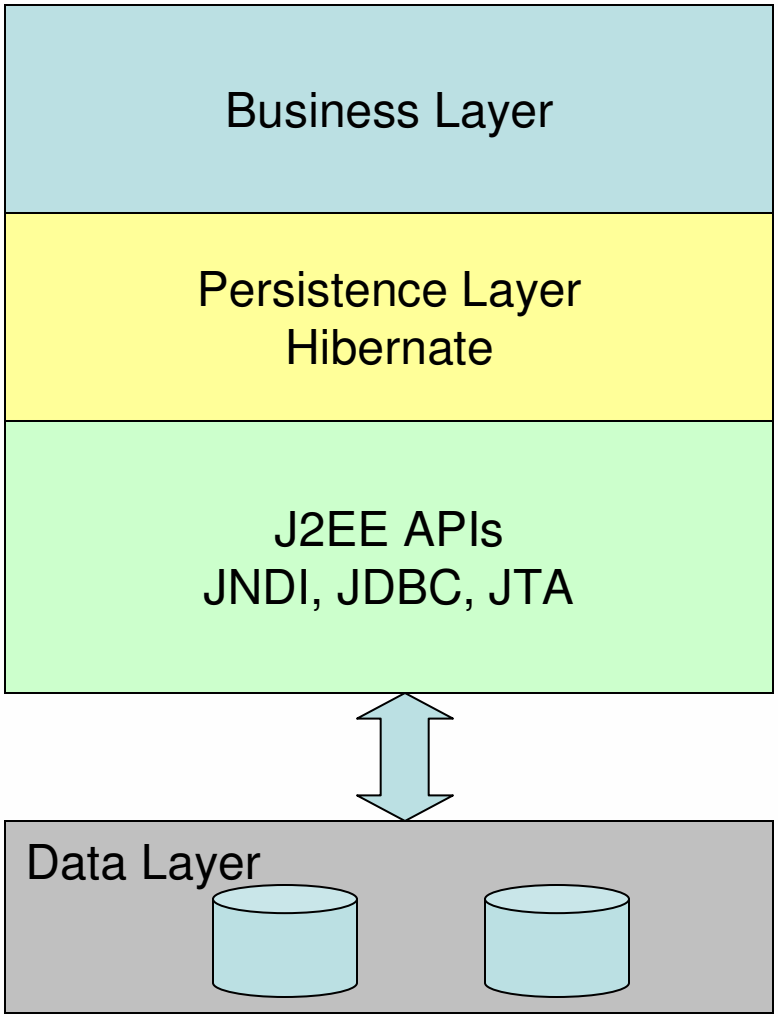
- HQL is fully object oriented query language

```
Session newSession = getSessionFactory().openSession();
Transaction newTransaction = newSession.beginTransaction();
List trades = newSession.find("from Trades as t order by t.tradeId asc");
System.out.println( trades.size() + " trades(s) found:" );
for ( Iterator iter = trades.iterator(); iter.hasNext(); ) {
        Trade trade = (Trade) iter.next();
        System.out.println( “ID: “ + trade.getOrderId() +
                        “ Symbol:” + trade.getSymbol() );
}
newTransaction.commit();
newSession.close();
```

Generates the below SQL Statement:

```
select t.TRADE_ID, t.CLIENT_ID, t.SYMBOL, t.ORDER_TYPE
        from TRADE t
        order by t.TRADE_ID asc
```

Hibernate Architecture



Hibernate Features

- Inheritance, Polymorphism Support
- Custom Data Types
- Collections
- Uni and Bi-directional entity Associations
- Transactions and concurrency
- Caching
- Connection Pooling
- HQL – Advanced Object Query Language etc

Hibernate Vs Others

- Other popular ORMs are
  - iBatis
  - JPA
  - TopLink
- iBatis
  - Needs SQL Statements to be coded in its Mapping files
  - Good when developer needs control over the SQL
- TopLink
  - Very similar and quite powerful but costs
  - Vendor lockin

Hibernate Vs Others cont..

- JPA – Java Persistence API
  - Java EE 5 ORM Solution
  - Part of EJB 3 Specification
  - Supported by all Java EE vendors
  - Designed based on popular ORM solutions like iBatis, JDO, TopLink including Hibernate
  - Replaces Entity Beans
  - It's a more of specification; you can use any provider like TopLink etc
  - Depends on provider which may implement more than standard specification
  - JPA lags in defining Caching and other advanced features
  - Useful in case of standard Java based solution using Java EE platform

Thank You

khaderv@yahoo.com

www.ksvali.com