LECTURE 13: OBLECT RELATIONAL MAPPING (ORM)

COMP2004J: Databases and Information Systems

Dr. Ruihai Dong (ruihai.dong@ucd.ie)
UCD School of Computer Science
Beijing-Dublin International College

What is ORM

When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.

ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc.

Hibernate

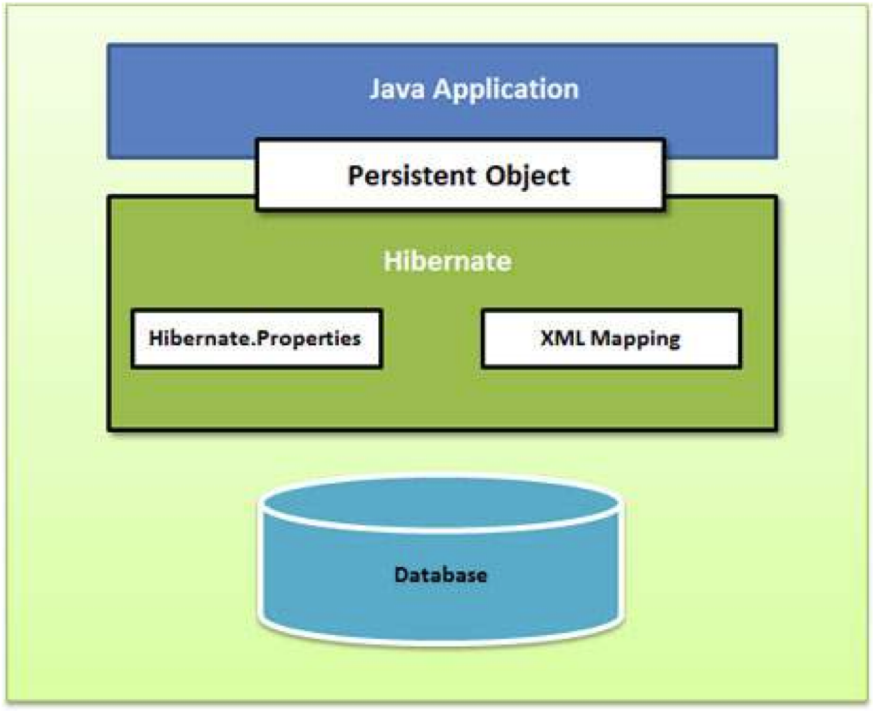
Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework created by Gavin King in 2001.

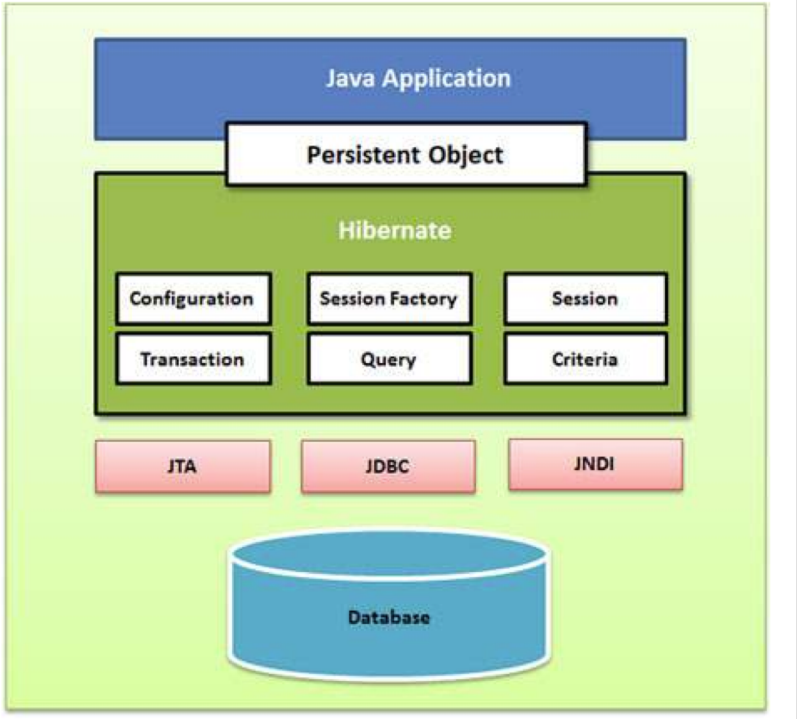
Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

Supported Databases

- HSQL Database Engine
- DB2/NT
- MySQL
- PostgreSQL
- FrontBase
- Oracle
- Microsoft SQL Server Database
- Sybase SQL Server
- Informix Dynamic Server

Hibernate Architecture





Persistent Class

The entire concept of Hibernate is to take the values from Java class attributes and persist them to a database table. A mapping document helps Hibernate in determining how to pull the values from the classes and map them with table and associated fields.

Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate. Hibernate works best if these classes follow some simple rules, also known as the **P**lain **O**ld **J**ava **O**bject (**POJO**) programming model.

POJO Example

```java
package hbm;

public class Department {

    private int deptno;
    private String deptname;
    private String office;
    private String division;
    private int managerno;


    public int getDeptno() {
        return deptno;
    }
    public void setDeptno(int deptno) {
        this.deptno = deptno;
    }
    public String getDeptname() {
        return deptname;
    }
    public void setDeptname(String deptname) {
        this.deptname = deptname;
    }
    public String getOffice() {
        return office;
    }
    public void setOffice(String office) {
        this.office = office;
    }
    public String getDivision() {
        return division;
    }
    public void setDivision(String division) {
        this.division = division;
    }
    public int getManagerno() {
        return managerno;
    }
    public void setManagerno(int managerno) {
        this.managerno = managerno;
    }
}
```

Mapping files

An Object/relational mappings are usually defined in an XML document. This mapping file instructs Hibernate how to map the defined class or classes to the database tables

Though many Hibernate users choose to write the XML by hand, a number of tools exist to generate the mapping document. These include **XDoclet**, **Middlegen** and **AndroMDA** for advanced Hibernate users

Mapping files

```xml
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="hbm">

    <class name="Department" table="department">
        <id name="deptno" column="deptno"/>
        <property name="deptname"/>
        <property name="office"/>
        <property name="division"/>
        <property name="managerno" column="managerno" type="int"/>
    </class>

</hibernate-mapping>
```

Configuration

Hibernate requires to know in advance where to find the mapping information that defines how your Java classes relate to the database tables.

Hibernate also requires a set of configuration settings related to database and other related parameters.

All such information is usually supplied as an XML file named **hibernate.cfg.xml**

hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost/employee</property>
        <property name="connection.username">root</property>
        <property name="connection.password">damocandid</property>

        <mapping resource="hbm/Department.hbm.xml"/>

    </session-factory>

</hibernate-configuration>
```

Create an Department in the Database

```
public void saveDepartments(Department d){
    Session session = factory.openSession();
    Transaction tx = null;
    try{
        tx = session.beginTransaction();
        session.save(d);
        tx.commit();
    }
    catch(HibernateException e){

    }
    finally{
        session.close();
    }
}
```

List All Departments

```
public void listDepartments(){
    Session session = factory.openSession();
    Transaction tx = null;
    try{
        tx = session.beginTransaction();
        List departs = session.createQuery("FROM Department").list();

        for(Object o:departs){
            Department d = (Department)o;
            System.out.println(d.getDeptname());
            System.out.println(d.getDeptno());
        }
        tx.commit();
    }
    catch(HibernateException e){

    }
    finally{
        session.close();
    }
}
```