Hibernate Tutorial



It was started in 2001 by Gavin King as an alternative to EJB2 style entity bean.

Hibernate Framework

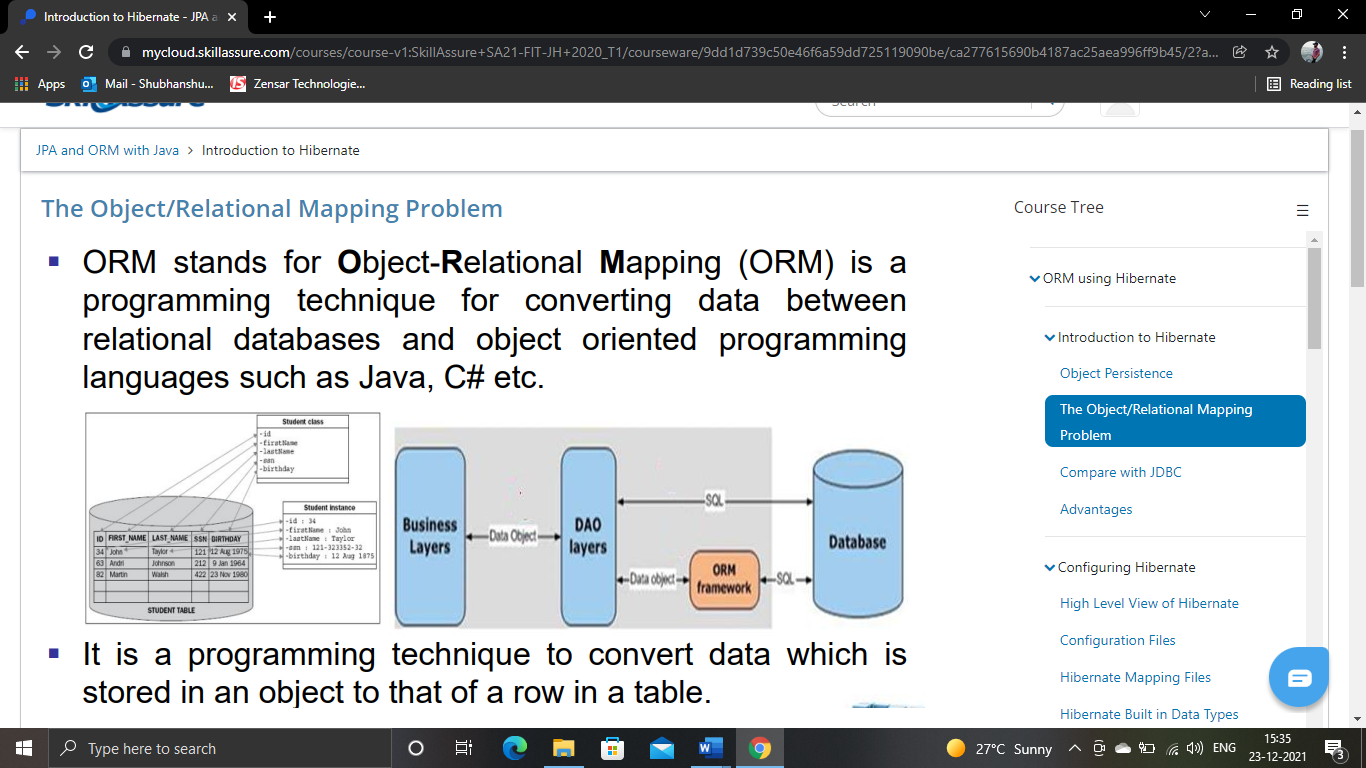
Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



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



What is JPA?

Java Persistence API (JPA) is a Java specification that provides certain functionality and standard to ORM tools. The **javax.persistence** package contains the JPA classes and interfaces.

Advantages of Hibernate Framework

Following are the advantages of hibernate framework:

1) Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

2) Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic Table Creation

Hibernate framework provides the facility to create the tables of the database automatically. So, there is no need to create tables in the database manually.

5) Simplifies Complex Join

Fetching data from multiple tables is easy in hibernate framework.

6) Provides Query Statistics and Database Status

Hibernate supports Query cache and provide statistics about query and database status.

# Hibernate Architecture

1. [Hibernate Architecture](https://www.javatpoint.com/hibernate-architecture)
2. [Elements of Hibernate Architecture](https://www.javatpoint.com/hibernate-architecture#elements)
   1. [SessionFactory](https://www.javatpoint.com/hibernate-architecture#e1)
   2. [Session](https://www.javatpoint.com/hibernate-architecture#e2)
   3. [Transaction](https://www.javatpoint.com/hibernate-architecture#e3)
   4. [ConnectionProvider](https://www.javatpoint.com/hibernate-architecture#e4)m
   5. [TransactionFactory](https://www.javatpoint.com/hibernate-architecture#e5)

The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand api layer
* Database layer

Let's see the diagram of hibernate architecture:



This is the high-level architecture of Hibernate with mapping file and configuration file.

4.7M

125

C++ vs Java



Hibernate framework uses many objects such as session factory, session, transaction etc. alongwith existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface).

## Elements of Hibernate Architecture

|  |
| --- |
| For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows: |

#### **SessionFactory**

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

#### **Session**

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

#### **Transaction**

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

#### **ConnectionProvider**

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

#### **TransactionFactory**

It is a factory of Transaction. It is optional.

SQL Dialects in Hibernate

The dialect specifies the type of database used in hibernate so that hibernate generate appropriate type of SQL statements. For connecting any hibernate application with the database, it is required to provide the configuration of SQL dialect.

Syntax of SQL Dialect

1. <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

List of SQL Dialects

There are many Dialects classes defined for RDBMS in the **org.hibernate.dialect** package. They are as follows:

|  |  |
| --- | --- |
| **RDBMS** | **Dialect** |
| Oracle (any version) | org.hibernate.dialect.OracleDialect |
| Oracle9i | org.hibernate.dialect.Oracle9iDialect |
| Oracle10g | org.hibernate.dialect.Oracle10gDialect |
| MySQL | org.hibernate.dialect.MySQLDialect |
| MySQL with InnoDB | org.hibernate.dialect.MySQLInnoDBDialect |
| MySQL with MyISAM | org.hibernate.dialect.MySQLMyISAMDialect |
| DB2 | org.hibernate.dialect.DB2Dialect |
| DB2 AS/400 | org.hibernate.dialect.DB2400Dialect |
| DB2 OS390 | org.hibernate.dialect.DB2390Dialect |
| Microsoft SQL Server | org.hibernate.dialect.SQLServerDialect |
| Sybase | org.hibernate.dialect.SybaseDialect |
| Sybase Anywhere | org.hibernate.dialect.SybaseAnywhereDialect |
| PostgreSQL | org.hibernate.dialect.PostgreSQLDialect |
| SAP DB | org.hibernate.dialect.SAPDBDialect |
| Informix | org.hibernate.dialect.InformixDialect |
| HypersonicSQL | org.hibernate.dialect.HSQLDialect |
| Ingres | org.hibernate.dialect.IngresDialect |
| Progress | org.hibernate.dialect.ProgressDialect |
| Mckoi SQL | org.hibernate.dialect.MckoiDialect |
| Interbase | org.hibernate.dialect.InterbaseDialect |
| Pointbase | org.hibernate.dialect.PointbaseDialect |
| FrontBase | org.hibernate.dialect.FrontbaseDialect |
| Firebird | org.hibernate.dialect.FirebirdDialect |

# JPA vs. Hibernate

## What is JPA?

A JPA (Java Persistence API) is a specification of Java which is used to access, manage, and persist data between Java object and relational database. It is considered as a standard approach for Object Relational Mapping.

JPA can be seen as a bridge between object-oriented domain models and relational database systems. Being a specification, JPA doesn't perform any operation by itself. Thus, it requires implementation. So, ORM tools like Hibernate, TopLink, and iBatis implements JPA specifications for data persistence.

## What is Hibernate?

A Hibernate is a Java framework which is used to store the Java objects in the relational database system. It is an open-source, lightweight, ORM (Object Relational Mapping) tool.

Hibernate is an implementation of JPA. So, it follows the common standards provided by the JPA.

## Need of JPA

As we have seen so far, JPA is a specification. It provides common prototype and functionality to ORM tools. By implementing the same specification, all ORM tools (like Hibernate, TopLink, iBatis) follows the common standards. In the future, if we want to switch our application from one ORM tool to another, we can do it easily.

## JPA vs. Hibernate

|  |  |
| --- | --- |
| **JPA** | **Hibernate** |
| Java Persistence API (JPA) defines the management of relational data in the Java applications. | Hibernate is an Object-Relational Mapping (ORM) tool which is used to save the state of Java object into the database. |
| It is just a specification. Various ORM tools implement it for data persistence. | It is one of the most frequently used JPA implementation. |
| It is defined in **javax.persistence** package. | It is defined in **org.hibernate** package. |
| The **EntityManagerFactory** interface is used to interact with the entity manager factory for the persistence unit. Thus, it provides an entity manager. | It uses **SessionFactory** interface to create Session instances. |
| It uses **EntityManager** interface to create, read, and delete operations for instances of mapped entity classes. This interface interacts with the persistence context. | It uses **Session** interface to create, read, and delete operations for instances of mapped entity classes. It behaves as a runtime interface between a Java application and Hibernate. |
| It uses **Java Persistence Query Language** (JPQL) as an object-oriented query language to perform database operations. | It uses **Hibernate Query Language** (HQL) as an object-oriented query language to perform database operations. |