

# Hibernate: Object-Relational Mapping Framework

## Introduction to ORM

- Fundamentally, applications written in Java perform logic. The Java language provides facilities for performing iterative logic with loops, conditional logic with if statements and object-oriented analysis through the use of classes and interfaces. But Java applications do not store data persistently.
- Data persistence is typically delegated to relational databases such as open source database MySQL or IBM's DB2 or Microsoft's SQL Server or NoSQL databases such as MongoDB and Cassandra.
- To address the object-relational impedance mismatch, a number of frameworks exist that simplify the task of moving data between a relational database and a Java program, some popular object-relational mapping (ORM) frameworks are Hibernate, TopLink and DataNucleus etc.. While each framework has its own set of unique capabilities, all of them comply with the Java Persistence API standard, which is now part of the Java EE/Jakarta EE specification.

## Object Relational Mapping(ORM)

- Object-Relational Mapping (ORM) is the process of converting Java objects to database tables. In other words, this allows us to interact with a relational database without any SQL.
- The Java Persistence API (JPA) is a specification that defines how to persist data in Java applications. The primary focus of JPA is the ORM layer.
- Hibernate is one of the most popular Java ORM framework in use today. Additionally, Hibernate is a standard implementation of the JPA specification, with a few additional features that are specific to Hibernate.



# Hibernate: Object-Relational Mapping Framework

## Introduction to ORM

- Fundamentally, applications written in Java perform logic. The Java language provides facilities for performing iterative logic with loops, conditional logic with if statements and object-oriented analysis through the use of classes and interfaces. But Java applications do not store data persistently.
- Data persistence is typically delegated to relational databases such as open source database MySQL or IBM's DB2 or Microsoft's SQL Server or NoSQL databases such as MongoDB and Cassandra.
- To address the object-relational impedance mismatch, a number of frameworks exist that simplify the task of moving data between a relational database and a Java program, some popular object-relational mapping (ORM) frameworks are Hibernate, TopLink and DataNucleus etc.. While each framework has its own set of unique capabilities, all of them comply with the Java Persistence API standard, which is now part of the Java EE/Jakarta EE specification.

## Object Relational Mapping(ORM)

- Object-Relational Mapping (ORM) is the process of converting Java objects to database tables. In other words, this allows us to interact with a relational database without any SQL.
- The Java Persistence API (JPA) is a specification that defines how to persist data in Java applications. The primary focus of JPA is the ORM layer.
- Hibernate is one of the most popular Java ORM framework in use today. Additionally, Hibernate is a standard implementation of the JPA specification, with a few additional features that are specific to Hibernate.



## Why use Hibernate?

Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

## Hibernate and JPA vs. JDBC

Java Database Connectivity (JDBC) is an API packaged with the Java SE edition that makes it possible to standardize and simplify the process of connecting Java applications to external, relational database management systems (RDBMS).

### Hibernate vs. JDBC comparison

TOOL	HIBERNATE JPA IMPLEMENTATION	JDBC
Purpose	Object-relational mapping (ORM)	Database Connectivity
Query Language	Hibernate Query Language (HQL) and the Java Persistence Query Language (JPQL)	Structured Query Language (SQL)
License	LGPL 2.1 and ASL 2.0	Oracle license
Packaging	Standalone JAR file along with dependent libraries	Part of Java SE
Caching	Built-in second-level caching	No caching
Competitors	TopLink, EclipseLink, OpenJPA and ActiveJDBC	ODBC, JDO
Maintainer	JBoss by Red Hat	Oracle
Release	Initial Hibernate release in 2001; JPA 1.0 released in 2006	Released in 1997 as part of Java 1.1

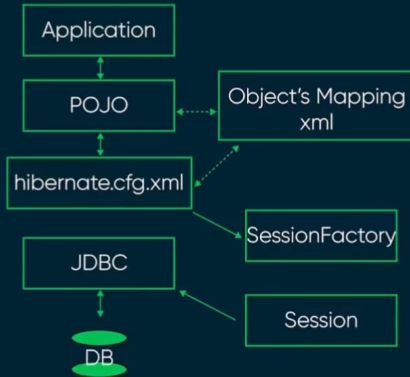


Hibernate is an open-source Java persistence framework. It performs powerful object-relational mapping and query databases using HQL and SQL.

Hibernate is a great tool for ORM mappings in Java. It can cut down a lot of complexity and thus defects as well from your application, which may otherwise find a way to exist. This is especially boon for developers with limited knowledge of SQL.

Initially started as an ORM framework, Hibernate has spun off into many projects, such as Hibernate Search, Hibernate Validator, Hibernate OGM (for NoSQL databases), and so on.

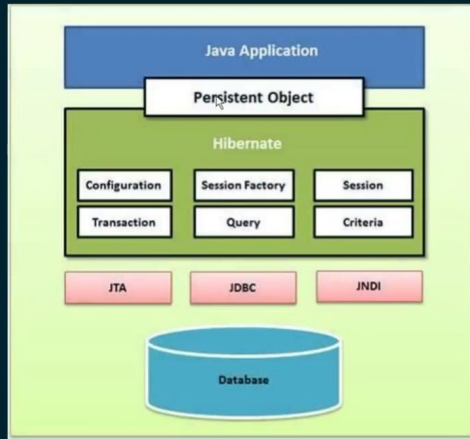
### Hibernate Architecture



## 5 Rules for generating POJO (Transfer Object) class)

1. All data members must be private.
2. Must implement serializable interface
3. Must have default constructor
4. Must have getter/setter methods  
(Accessor/Mutator Methods)
5. Class must be non-final





## Configuration:

- Configuration is a class which is present in org.hibernate.cfg package. It activates Hibernate framework. It reads both configuration file and mapping files.
- It activate Hibernate Framework
- Configuration cfg=new Configuration();
- It read both cfg file and mapping files
- cfg.configure();
- It checks whether the config file is syntactically correct or not.
- If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.

## SessionFactory:


- SessionFactory is an Interface which is present in org.hibernate package and it is used to create Session Object.
- It is immutable and thread-safe in nature.
- buildSessionFactory() method gathers the meta-data which is in the cfg Object.
- From cfg object it takes the JDBC information and create a JDBC Connection.
- SessionFactory factory=cfg.buildSessionFactory();

## Session:

- Session is an interface which is present in org.hibernate package. Session object is created based upon SessionFactory object i.e. factory.
- It opens the Connection/Session with Database software through Hibernate Framework.
- It is a light-weight object and it is not thread-safe.
- Session object is used to perform CRUD operations.
- Session session=factory.buildSession();



The Session interface defines a number of methods for executing CRUD operations with mapped objects. The following table lists the most common methods provided by the session:

#	Method name	Return	Description	Issued SQL statement
1	beginTransaction() 	Transaction	Creates a Transaction object or returns an existing one, for working under context of a transaction.	
2	getTransaction()	Transaction	Returns the current transaction.	
3	get(Class class, Serializable id)	Object	Loads a persistent instance of the given class with the given id, into the session.	SELECT
4	load(Class class, Serializable id)	Object	Does same thing as get() method, but throws an ObjectNotFound error if no row with the given id exists.	SELECT
5	persist(Object)	void	saves a mapped object as a row in database	INSERT
6	save(Object)	Serializable	Does same thing as persist() method, plus returning a generated identifier.	INSERT
7	update(Object)	void	Updates a detached instance of the given object and the underlying row in database.	UPDATE
8	saveOrUpdate(Object)	void	Saves the given object if it does not exist, otherwise updates it.	INSERT or UPDATE
9	delete(Object)	void	Removes a persistent object and the underlying row in database.	DELETE
10	close()	void	Ends the current session.	
11	flush()	void	Flushes the current session. This method should be called before committing the transaction and closing the session.	
12	disconnect()	void	Disconnects the session from current JDBC connection.	





### Transaction:

- Transaction object is used whenever we perform any operation and based upon that operation there is some change in database.
- Transaction object is used to give the instruction to the database to make the changes that happen because of operation as a permanent by using `commit()` method.
- `Transaction tx=session.beginTransaction();`
- `tx.commit();`

### Query:

- Query is an interface that present inside `org.hibernate` package.
- A Query instance is obtained by calling `Session.createQuery()`.
- This interface exposes some extra functionality beyond that provided by `Session.iterate()` and `Session.find()`:
  1. A particular page of the result set may be selected by calling `setMaxResults()`, `setFirstResult()`.
  2. Named query parameters may be used.
- `Query query=session.createQuery();`

### Criteria:

- Criteria is a simplified API for retrieving entities by composing Criterion objects.
- The Session is a factory for Criteria. Criterion instances are usually obtained via the factory methods on Restrictions.
- `Criteria criteria=session.createCriteria();`



## Creating Hibernate Project using Eclipse IDE

Here, we are going to create a simple example of hibernate application using eclipse IDE. For creating the first hibernate application in Eclipse IDE, we need to follow the following steps:

1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file
6. Create the class that retrieves or stores the persistent object
7. Run the application

### 1. Create the java project

Create the java project by File Menu - New - project - java project . Now specify the project name e.g. firsthb then next - finish.



## **Hibernate Query Language (HQL)**

Hibernate provides its query language called Hibernate Query Language (HQL). HQL is an object-oriented query language, similar to the native SQL language, and it works with persistent objects

### **Advantages of HQL**

- Database Independent- HQL is a database-independent query language. That means if we write programs using HQL commands, then the program can be executed in all relational databases without any modification.



@Entity

@Table

@Column

@Id

@Generic  
Generator

@Generated  
Value



# Hibernate Relationship

Using hibernate, if we want to put relationship between two entities [ objects of two pojo classes ], then in the database tables, there must exist foreign key relationship, we call it as Referential integrity.

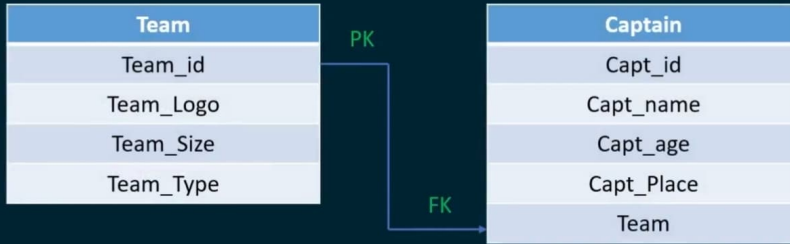
Using hibernate we can put the following 4 types of relationships

1. One-To-One
2. One-To-Many
3. Many-To-One
4. Many-To-Many



## 1. One-To-One Relationship

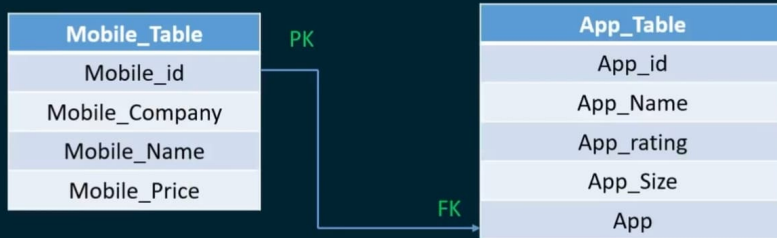
Either end can be made by the owner, but one (and only one) of them should be; if we don't specify this, we will end up with a circular dependency.



## 2. One-To-Many Relationship

While adding a @OneToMany relationship is very easy with JPA and Hibernate, knowing the right way to map such an association so that it generates very efficient SQL statements is definitely not a trivial thing to do.

In a relational database system, a one-to-many association links two tables based on a Foreign Key column so that the child table record references the Primary Key of the parent table row.



# Hibernate Caching/Cache

Caching is a mechanism to enhance the performance of a system. It is a buffer memory that lies between the application and the database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.





## Hibernate Caching/cache

user->server->db

```
user1=select * from student;->Tomcat  
session s1=s1.get(1);
```

```
user2=select * from student;  
session s2=s2.get(1);
```

In order to provide second level cache it will provide the third party libraries.

1. Ehcache
2. OS cache
3. Swarm cache

### Ehcache

->First thing

->Inorder to get ehcache you have to download the library called ehcache.jar

->Also download jar file called hibernate-ehcache.jar file

### Ehcache provide the Features

hibernate-ehcache.jar provides integration of eh and hibernate

### Second thing

->You have to configure hibernate.cfg.xml

->By default your second level cache is disable so you have to enable it by using <>.

### Third thing

->You have to change the entity by using annotation

@Cacheable

@Cache

## Hibernate Caching/Cache

