## Hibernate:

* Hibernate is an ORM tool. that simplify the development of Java application to interact with the database.
* Hibernate develops persistence logic, which stores and processes the data for longer use.
* It is a lightweight tool and most importantly **open-sourced**

# Advantages:

* open-sourced and lightweight.
* very fast.
* Helps in generating database independent queries.
* Provides facilities to automatically create a table.
* It provides query statistics and database status.

# Hibernate Over JDBC:

* Remove Boiler Plate Code
* Support **Association**, **Inheritance** and **collection**.
* Support HQL Query.
* Implicit Transaction Management
* Throw unchecked Exception

# ORM

* Object Relational Mapping
* Maps the object which are stored in database
* simplify creation manipulation and access.

# HQL

* HQL stands for hibernate query language
* It is object oriented query language , independent of database.

# Session VS Session Factory:

* **sessionFactory**:
* SessionFactory is used to get session .
* sessionFactory is thread safe.
* **session**
* session is used to maintain connection with database.
* is not thread safe

# getCurrentSession() VS openSession():

* **getCurrentSession**:
* it return session
* it automatically close when we close sessionFactory.
* **openSession**:
* it return session
* it will not automatically close when we close sessionFactory.

# Key Components of Hibernate:

* **Database Connection**: managed by one or more config file
* **Class Mapping**: used to map your pojo with database.

# Hibernate Template:

* It is used to perform transaction management from spring transaction.
* Automatically close session.

# Design Pattern Used By Hibernate:

* **Domain Model**: it incorporates the data and ports.
* **Data Mapper** : it creates a layer in which data maps b/w database and object class when they are independent
* **Proxy Pattern** : it is used for lazy loading
* **Factory Pattern** : it creates session factory and keep track of each session

# Validator Framework:

* It is used in presentation layer used to perform validation with the help of JavaScript and server side

# Light Object Mapping:

* Entity classes mapped manually to database tables.
* Business logic code is hidden from data access code
* Application with less number of entities use this concept..

# Hibernate Tuning:

* SQL Optimization
* Session Management
* Data Caching.

# Hibernate States:

* **Transient**: object is not associated with session
* **Persistent**: associated with session
* **Detached**: when session close persistent state become detached

# Proxy & Lazy Loading:



# Query Level Cache:

* it create separates cache region for each query
* it is used when query having same parameters.

# Merge & Update:

* **Merge**:
* it can do modification any time without considering the state of the session.
* **Update**:
* it is used when session doesn't contain any persistent instance with same id.

# Get VS Load:

* **GET**
* Get will return null
* get will not return proxy object
* **Load**
* Load will return ObjectNotFoundException
* Load will return proxy object

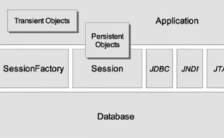
# First Level Caching VS Second Level Caching:

* **First Level Cache**
* It is enabled by default
* When you request single data from two different session then query will be fired twice and hit database twice.
* First Level cache is associated with session.
* **Second Level Cache**
* It is not enabled by default.
* It is associated with session Factory
* when you request single data from two different session then query will be fired once and hit database only once.

# JPA VS Hibernate:

* **JPA**:
* JAVA persistence API is the standard specification of orm
* it doesn't support second level cache.
* it support .net using NHibernate tool
* it generated JPQL.
* **Hibernate**:
* Hibernate is the actual implementation of orm
* it support second level cache.
* it doesn't support .net using NHibernate tool
* it generated HQL

## Hibernate Architecture:



# Hibernate Unique Identifier Generators:

* **Identity** : generate unique identity in database mysql,sql.
* **Sequence**: generate database named sequence for generating identifiers
* **Auto**: use on of above based on database.

## Optimistic Locking:

* in case of log running parallel transaction it provides required scalability
* without it table will be locked , avoiding concurrent access.
* it can be achieved by @Version annotation.
* if something is changes then this column will be updated .

# N+1 Select Problem :

* after association of parent child relationship at the time of data retrieving hibernate execute **N+1** query one for parent and N for child .
* for avoiding it we should use **select fetchMode join**.

# Schema Export:

* Tool which is used to automatic generate schema with all mapping.

# Key Components:

* Configuration
* SessionFactory
* Session
* Transaction
* Query
* Criteria

# Configuration:

* Usually created once during application initialization
* Hibernate Object you create in any application.

# Save VS Persist:

* save return id of the instance where persist doesn't return anything.

# Concurrency Strategies:

* **Transactional**: it mainly used to read data where it critical to prevent stale data for concurrent transaction
* **Read-write**: it is used when we perform alot of updates.
* **Nonstrict-read-write**: it prevent database consistency and used we have minimal updates.
* **Read-only**: we can insert and retrieve data but can't read it.