Protocol:-It transfer data from one place to another place.

Connection Properties:-It is information which required to connect to the database

1. Driver Class 🡪 com.mysql.cj.jdbc.Driver
2. Url 🡪 jdbc:mysql://localhost:3306/resturantinfo
3. Username 🡪root
4. Password 🡪tiger

Configuration:-It certain use case connection to the database

Configuration Connection Properties

Mapping Class (Annnoted class)

Refrence of session factory

Connection DB

Seassion Factory Managing Configuration

Refrence of session

DB

Session Factory

Session

Configuration

1. Configuration DB
2. ManagingDb Configuration
3. Refrence of session(sessionFactory.openSession()
4. Configuration properties(configuration.setProperties(properties);
5. Mapping Class(addAnnotatedClass(RestaurantInfo.class)
6. Refrence of session factory(configuration.buildSessionFactory()
7. Live Connection
8. DB Operation
9. Session(object obj)

A

**Trasaction:-**

**Atomicity (A):** This property ensures that a transaction is treated as a single, indivisible unit of work. Either all the changes made in a transaction are committed to the database or none of them are.

**Consistency (C):** This property ensures that a transaction brings the database from one consistent state to another. If the database is in a consistent state before a transaction starts, it must be in a consistent state when the transaction is finished.

**Isolation (I):** This property ensures that the changes made by one transaction are isolated from the changes made by other transactions. This means that multiple transactions can be executed concurrently without interfering with each other.

**Durability (D):** This property ensures that the changes made to the database by a transaction are permanent, even in the case of system failures such as power outages or crashes. The changes made by a committed transaction must persist in the database.

Annnotation:-

1. @Entity
2. @Table(name =EntityConstant.***RESTAURANT\_INFO***)
3. @GenericGenerator(name="auto" , strategy = "increment")
4. @GeneratedValue(generator = "auto")

Inbuild Method:

1. Save(object obj)
2. Update(object obj)
3. Get(.class,id)
4. Delete(object obj)

HQL-Hibernate Query Language:-

It is depends upon your entity class object ,it is does not depend upon your databasebase table.

SQL:-select \* from resturat\_info;

HQL:-select \* from ResturantInfo;

SQL:- select name,pin\_code form resturant\_info;(tables attributes name/column name)

HQL:-select name,pincode from ResturantInfo;(entity class variable name)

**FrameWork:-**

Spring boot solution to common problems

Example: Spring Boot ,Angular,Hibernate,Spring,structs etc.

**Hibernate:-**

It is a framework which gives solution to jdbc or ORM problems.

**DTO:-( Data Transfer Object)**

It is used to transfer data from one class to another class.

1. Class should be a non final class
2. Class should be implements serializable interface
3. All the properties in the class should be private
4. Getter and Setter for all the properties

**DAO (Data Access object ) :-**

It is class which will having the database logic.

**ORM(object relational Mapping) :-**

ORM is used to persist the data into the relation database without converting object into values and values back to object.

**ORM problems:-**

1. Problem of inheritance
2. Problem of Relations
3. Problem of identity
4. Problem of grain
5. Problem navigation

**JPA (Java s. persistence)**

**It is set of starter ,timeline for the ORM problem.**

**Component:-**

* **Configuration**
  1. **It is used provides connection provides information**
  2. **It provides mapping class information**
  3. **It gives the reference of session factory.**
  4. **BuildSessionFactory():-This is it used to return the implementation of session factory**

**Session Factory:-**

1. **It is configure the data base for application .**
2. **It manages the database connection .**
3. **It gives the reference of session.**
4. **Opensession:-This method is used to return the implementation object of session.**

**Session:-**

1. **It is a live connection between a application and database.**
2. **It is used to perform the database operation like save, update, delete and get operation.**
3. **Session gives the reference of transaction , criteria, query etc.**

**Note:-**

1. **Session factory database connection object it is recommended to have one session factory for application.**
2. **From one session factory we can get multiple session.**

**Annotation**

**@Entity:-**

It is used to represent the java class which will mapped to a table inside the database.

An entity represents table inside the database.

**@Table:-**

It is allows to specify the tables details which can used to persist the entity to the database.

**@column:-**

It is used to map of property of entity class to a column inside the database.

**@Id:-**

This annotation is used to specify that property is primary key inside the table.

**HQL (Hibernate query language):-**

It is a way of representing the queries based on object model.

It is independed of database.

Note:-HQL is having the similar syntax as SQL but instead of table name we used entity class name and instead column name we use property name .

**Relationship:-**

1. **One-to-One (1:1) relationship:** In this relationship, each record in Table A is related to one and only one record in Table B, and vice versa. For example, an Employee table and a Login table where each employee has a unique login and each login belongs to a unique employee.

**Team** 🡪 Captain

**Company 🡪**CEO

**Person 🡪AdharCard**

**Person 🡪Heart**

**Country 🡪 PrimeMinsiter**

1. **One-to-Many (1:N) relationship:** In this relationship, one record in Table A is related to many records in Table B, but each record in Table B is related to only one record in Table A. For example, a Department table and an Employee table where each department has many employees, but each employee belongs to only one department.
2. **Many-to-One (N:1) relationship:** In this relationship, many records in Table A are related to one record in Table B, but each record in Table B is related to only one record in Table A. This is the reverse of a one-to-many relationship.
3. **Many-to-Many (N:M) relationship:** In this relationship, many records in Table A are related to many records in Table B and vice versa. This type of relationship is implemented through an additional junction table that defines the relationships between records in Table A and Table B. For example, an Employee table and a Project table, where each employee can work on many projects and each project can have many employees.

**Cache:-**

It is nothing but a buffer memory which is store some certain information.

Two types of cache

First level :-Session object/for each operation individual session/hibernate all time active.It is only available in particular method.

Second level :-