102010505

Advanced Java

Unit-5 Hibernate

Hibernate: Introduction

- Hibernate is used to convert object data in JAVA to relational database tables.
- It is an open source Object-Relational Mapping (ORM) for Java.
- Hibernate is responsible for making data persistent by storing it in a database.

Object-Relational Mapping (ORM)

- It is a programming technique for converting object-type data of an object oriented programming language into database tables.
- Hibernate is used to convert object data in JAVA to relational database tables.

JDBC v/s Hibernate

JDBC	Hibernate
JDBC maps Java classes to database tables (and from Java data types to SQL data types)	Hibernate automatically generates the queries.
With JDBC, developer has to write code to map an object model's data to a relational data model.	Hibernate is flexible and powerful ORM to map Java classes to database tables.
With JDBC, it is developer's responsibility to handle JDBC result set and convert it to Java. So with JDBC, mapping between Java objects and database tables is done manually.	Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects, hence reducing the development time and maintenance cost.

JDBC vs Hibernate

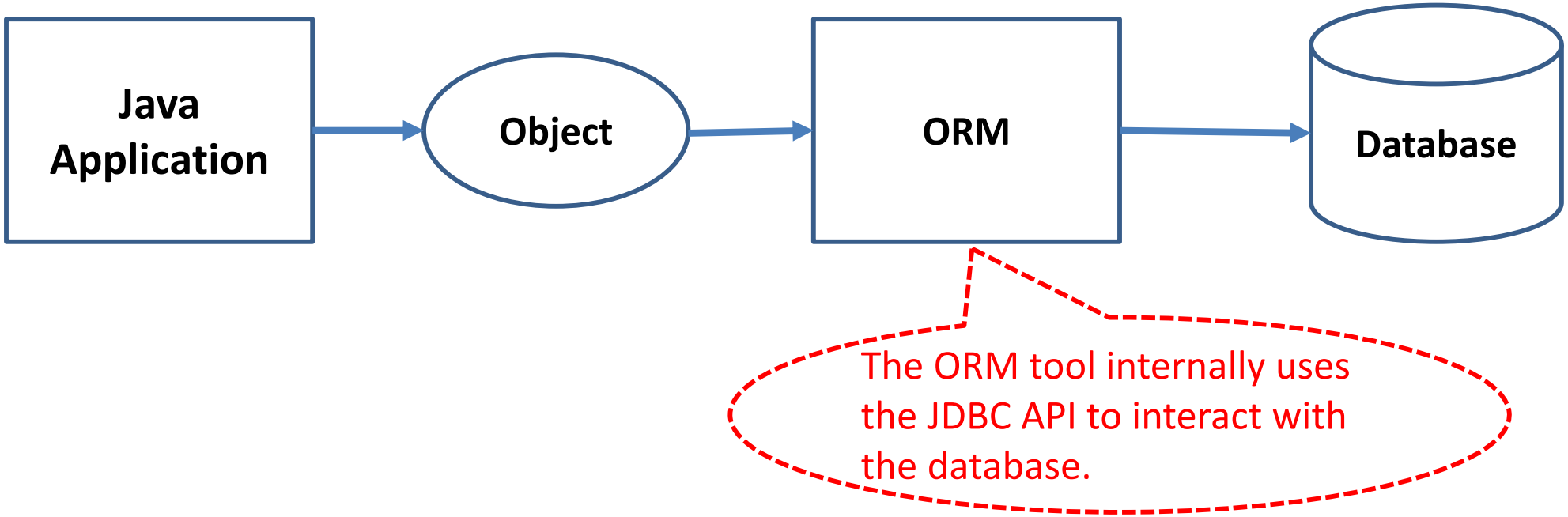
JDBC	Hibernate
Require JDBC Driver for different types of database.	Makes an application portable to all SQL databases.
Handles all create-read-update-delete (CRUD) operations using SQL Queries.	Handles all create-read-update-delete (CRUD) operations using simple API; no SQL
Working with both Object-Oriented software and Relational Database is complicated task with JDBC.	Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.
JDBC supports only native Structured Query Language (SQL)	Hibernate provides a powerful query language Hibernate Query Language-HQL (independent from type of database)

Overview of Hibernate

Hibernate Framework

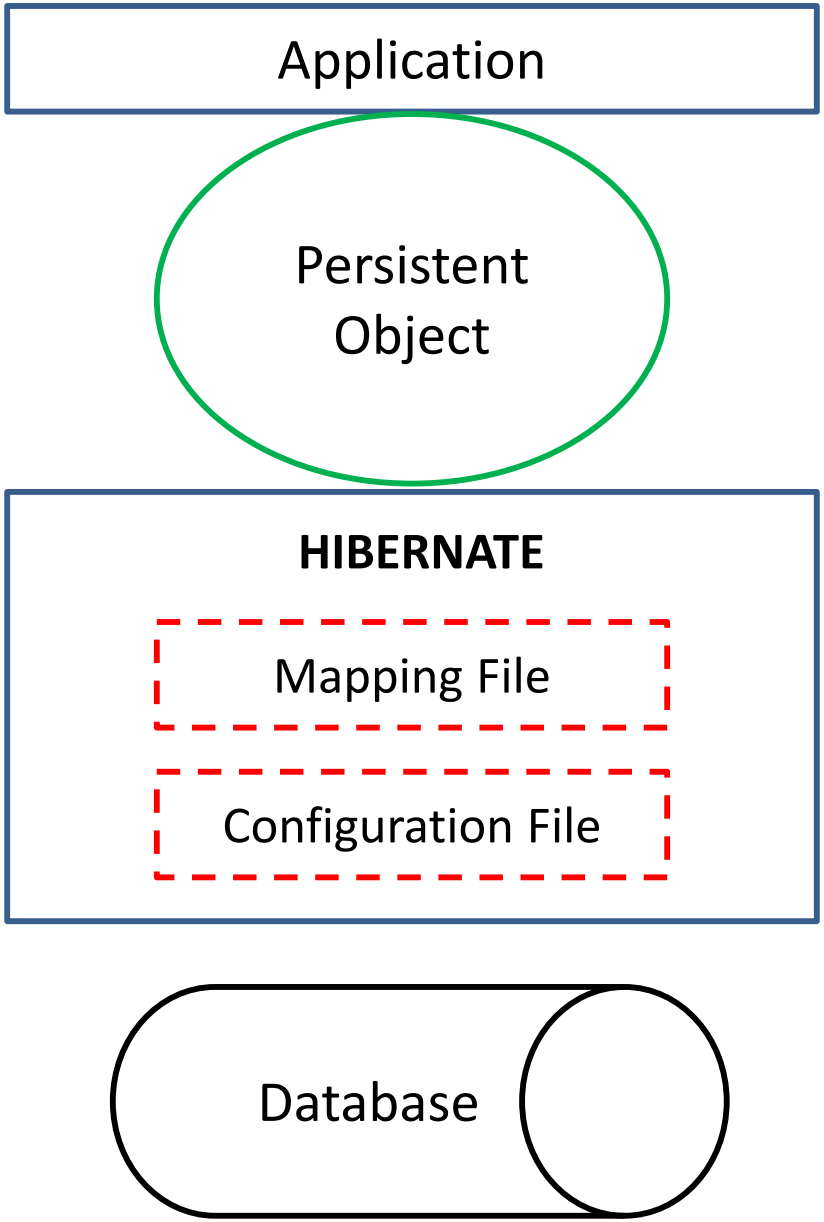
- Hibernate framework simplifies the development of java application to interact with the database.
- Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool.
- An ORM tool simplifies the data creation, data manipulation and data access.
- Hibernate is a programming technique that maps the object to the data stored in the database.

Hibernate Framework

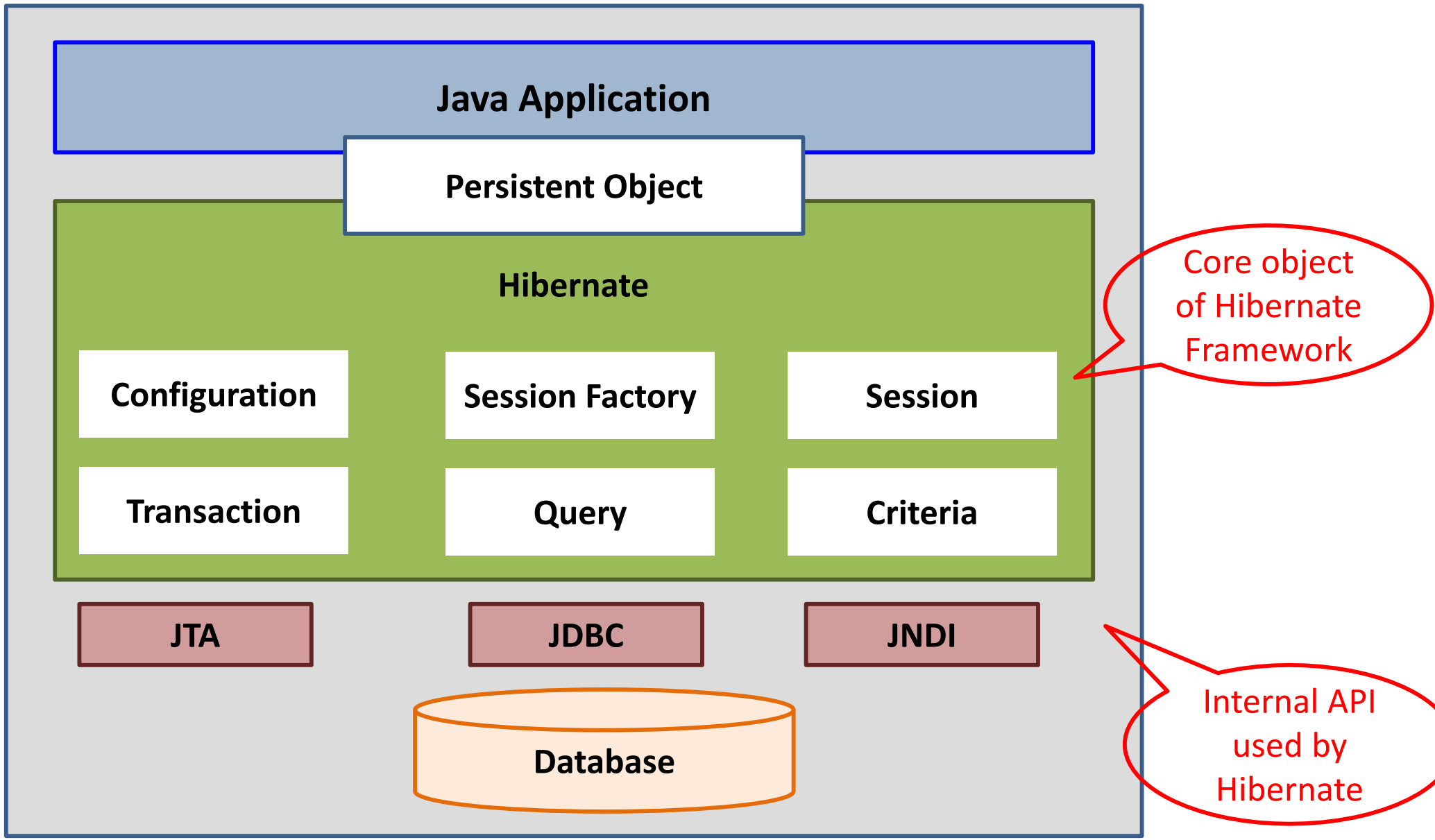


Hibernate Architecture

- There are 4 layers in hibernate architecture
  1. Java application layer
  2. Hibernate framework layer
  3. Backend API layer
  4. Database layer.



Hibernate Architecture



Hibernate Architecture

What do you mean by *Persistence*?

Persistence simply means that we would like our application's data to outlive the applications process. In Java terms, we would like the state of (some of) our objects to live beyond the scope of the JVM so that the same state is available later.

Hibernate Architecture

Internal API used by Hibernate

1. JDBC (Java Database Connectivity)
2. JTA (Java Transaction API)
3. JNDI (Java Naming Directory Interface)

https://www.youtube.com/watch?v=hfv9ZXUzjhk

Hibernate Architecture

- For creating the first hibernate application, we must know the objects/elements of Hibernate architecture.
- They are as follows:
  1. Configuration
  2. Session factory
  3. Session
  4. Transaction factory
  5. Query
  6. Criteria

Hibernate Architecture

[1] Configuration Object

- The Configuration object is the first Hibernate object you create in any Hibernate application.
- It is usually created only once during application initialization.
- The Configuration object provides two keys components:

1. **Database Connection:**

   This is handled through one or more configuration files supported by Hibernate. These files are **hibernate.properties** and **hibernate.cfg.xml**.

2. **Class Mapping Setup**:

   This component creates the connection between the Java classes and database tables.

Hibernate Architecture

[2] SessionFactory Object

- The SessionFactory is a thread safe object and used by all the threads of an application.
- Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application.
- You would need one SessionFactory object per database using a separate configuration file.
- So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Hibernate Architecture

[3] Session Object

- A Session is used to get a physical connection with a database.
- The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.
- The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed as needed.

Hibernate Architecture

[4] Transaction Object

- A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality.
- Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

Hibernate Architecture

[5] Query Object

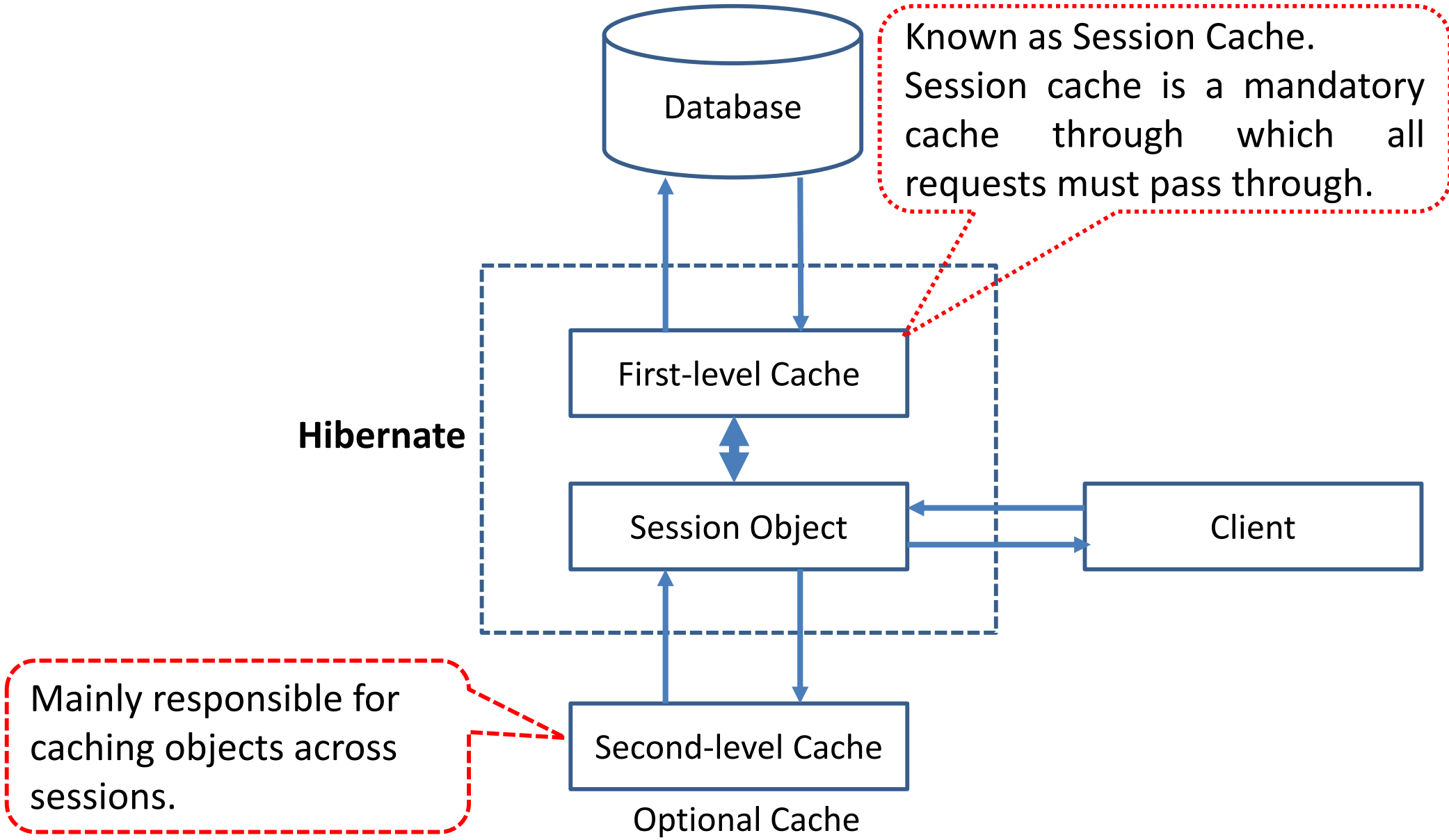
- Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.
- A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Hibernate Architecture

[6] Criteria Object

Criteria objects are used to create and execute object oriented criteria queries to retrieve objects.

Hibernate Cache Architecture



Why Cache Architecture?

- Caching is all about application performance optimization.
- It is situated between your application and the database to avoid the number of database hits as many as possible.
- To give a better performance for critical applications.

Hibernate Cache Architecture

First-level cache:

- The first-level cache is the Session cache.
- The Session object keeps an object under its own control before committing it to the database.
- If you issue multiple updates to an object, Hibernate tries to delay doing the update as long as possible to reduce the number of update SQL statements issued.
- If you close the session, all the objects being cached are lost.

Hibernate Cache Architecture

Second-level cache:

- It is responsible for caching objects across sessions.
- Second level cache is an optional cache and first-level cache will always be consulted before any attempt is made to locate an object in the second-level cache.
- Any third-party cache can be used with Hibernate. An **org.hibernate.cache.CacheProvider** interface is provided, which must be implemented to provide Hibernate with a handle to the cache implementation.

Hibernate Mapping Types

- While preparing a Hibernate mapping document, we map the Java data types into RDBMS data types.
- The types declared and used in the mapping files are not Java data types; they are not SQL database types either.
- These types are called ***Hibernate mapping types***, which can translate from Java to SQL data types and vice versa.

Hibernate Mapping Types:

Primitive Types

Mapping type	Java type	SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
character	java.lang.String	CHAR(1)
byte	byte	TINYINT
boolean	boolean	BIT
true/false	boolean	CHAR(1) ('T' or 'F')

Hibernate Mapping Types:

Date and time types:

Mapping type	Java type	SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Hibernate Mapping Types:

Binary and large object types:

Mapping type	Java type	SQL Type
binary	byte[]	BLOB
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable	BLOB
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB

Hibernate Mapping Types:

JDK-related types:

Mapping type	Java type	SQL Type
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
timezone	java.util.TimeZone	VARCHAR
currency	java.util.Currency	VARCHAR

Hibernate O/R Mapping

Three most important mapping are as follows:

1. Collections Mappings
2. Association Mappings
3. Component Mappings

Hibernate O/R Mapping

Collections Mappings

- If an entity or class has collection of values for a particular variable, then we can map those values using any one of the collection interfaces available in java.
- Hibernate can persist instances of **java.util.Map, java.util.Set, java.util.SortedMap, java.util.SortedSet, java.util.List**, and any **array** of persistent entities or values.

Hibernate O/R Mapping:

Collection type	Mapping and Description
java.util.Set	This is mapped with a **<set>** element and initialized with java.util.HashSet
java.util.SortedSet	This is mapped with a **<set>** element. The sort attribute can be set to either a comparator or natural ordering.
java.util.List	This is mapped with a <**list**> element and initialized with java.util.ArrayList
java.util.Collection	This is mapped with a <**bag**> or <**ibag**> element and initialized with java.util.ArrayList
java.util.Map	This is mapped with a <**map**> element and initialized with java.util.HashMap
java.util.SortedMap	This is mapped with a <**map**> element. The sort attribute can be set to either a comparator or natural ordering.

Hibernate O/R Mapping:

Association Mappings:

- The mapping of associations between entity classes and the relationships between tables is the soul of ORM.
- There are the four ways in which the cardinality of the relationship between the objects can be expressed.
- An association mapping can be unidirectional as well as bidirectional.

Hibernate O/R Mapping:

Association Mappings:

Mapping type	Description
Many-to-One	Mapping many-to-one relationship using Hibernate
One-to-One	Mapping one-to-one relationship using Hibernate
One-to-Many	Mapping one-to-many relationship using Hibernate
Many-to-Many	Mapping many-to-many relationship using Hibernate

Hibernate O/R Mapping:

Component Mappings:

- If the referred class does not have it's own life cycle and completely depends on the life cycle of the owning entity class, then the referred class hence therefore is called as the Component class.
- The mapping of Collection of Components is also possible in a similar way just as the mapping of regular Collections with minor configuration differences.

Mapping type	Description
Component Mappings	Mapping for a class having a reference to another class as a member variable.

Advantages of Hibernate Framework

1. **Open source and Lightweight:** Hibernate framework is open source under the LGPL (GNU Lesser General Public License) license and lightweight.
2. **Fast performance:** The performance of hibernate framework is fast because cache is internally used in hibernate framework.
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.

Advantages of Hibernate Framework

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:** To fetch 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 Query Language (HQL)

- The Hibernate ORM framework provides its own query language called Hibernate Query Language .
- Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL.

  Therefore, it is database independent query language.

Hibernate Query Language (HQL)

Characteristics of HQL

1. **Similar to SQL**

   HQL's syntax is very similar to standard SQL. If you are familiar with SQL then writing HQL would be pretty easy.

2. **Fully object-oriented:** HQL doesn't use real names of table and columns. It uses class and property names instead. HQL can understand inheritance, polymorphism and association.

3. **Reduces the size of queries**

HQL vs SQL

SELECT QUERY

SQL

ResultSet rs=st.executeQuery("**select * from diet**");

HQL

Query query= session.createQuery("**from diet**");

//here persistent class name is diet

HQL vs SQL

SELECT with WHERE clause

SQL

ResultSet rs=st.executeQuery("**select * from diet where id=301**");

HQL

Query query= session.createQuery("**from diet where id=301** ");

//here persistent class name is diet

HQL vs SQL

UPDATE QUERY

SQL

1. String query = "**update User set name=? where id = ?";**
2. PreparedStatement preparedStmt = conn.prepareStatement(query);
3. preparedStmt.setString (1, “DIET_CE”);
4. preparedStmt.setInt(2, 054);
5. preparedStmt.executeUpdate();

HQL

1. Query q=session.createQuery("**update User set name=:n where id=:i**");
2. q.setParameter("n", "DIET_CE");
3. q.setParameter("i",054);
4. int status=q.executeUpdate();

HQL vs SQL

INSERT QUERY

SQL

String sql = "**INSERT INTO Stock VALUES (100, 'abc')**";

int result = stmt.executeUpdate(sql);

HQL

Query query = session.createQuery("**insert into Stock(stock_code, stock_name) select stock_code, stock_name from backup_stock**");

int result = query.executeUpdate();

Steps to run hibernate example

Steps to run first hibernate example with MySQL in Netbeans IDE 8.2

Step-1: Create the database

```
CREATE DATABASE retailer;
```

Step-2: Create table result

```
CREATE TABLE customers(
        name varchar(20),
        C_ID int NOT NULL AUTO_INCREMENT,
        address  varchar(20),
        email varchar(50),
        PRIMARY KEY(C_ID)
);
```

Steps to run hibernate example

Step-3: Create new java application.

- File > New project > Java > Java Application > Next Name it as **HibernateTest.**
- Then click Finish to create the project.

Steps to run hibernate example

Step-4: Create a POJO(Plain Old Java Objects) class

- We create this class to use variables to map with the database columns.
- Right click the package (hibernatetest) & select New > Java Class Name it as Customer.
- Click Finish to create the class.

Steps to run hibernate example: Step-4

```
1. package hibernatetest;
2. public class Customer {
3.     public String customerName;
4.     public int customerID;
5.     public String customerAddress;
6.     public String customerEmail;
7. public void setCustomerAddress(String
                                    customerAddress) {
            this.customerAddress = customerAddress;}
8. public void setCustomerEmail(String customerEmail) {
9.          this.customerEmail = customerEmail;}
10.public void setCustomerID(int customerID) {
11.         this.customerID = customerID; }
```

Steps to run hibernate example: Step-4

```
12. public void setCustomerName(String customerName) {
13.         this.customerName = customerName;    }
14. public String getCustomerAddress() {
15.         return customerAddress;    }
16. public String getCustomerEmail() {
17.         return customerEmail;    }
18. public int getCustomerID() {
19.         return customerID;    }
20. public String getCustomerName() {
21.         return customerName;    }
22.}
```

Steps to run hibernate example: Step-4

Step-4: Create a POJO(Plain Old Java Objects) class

- To generate getters and setters easily in NetBeans, right click on the code and select Insert Code Then choose Getter... or Setter...
- Variable **customerName** will map with the name column of the **customers** table.
- Variable **customerID** will map with the **C_ID** column of the customers table. It is integer & auto incremented. So POJO class variable also should be int.
- Variable **customerAddress** will map with the **address** column of the customers table.
- Variable **customerEmail** will map with the **email** column of the customers table.

Steps to run hibernate example

Step-5: Connect to the database we have already created. [retailer]

- Select Services tab lying next to the Projects tab.
- Expand Databases.
- Expand MySQL Server. There we can see the all databases on MySQL sever
- Right click the database retailer. Select Connect.

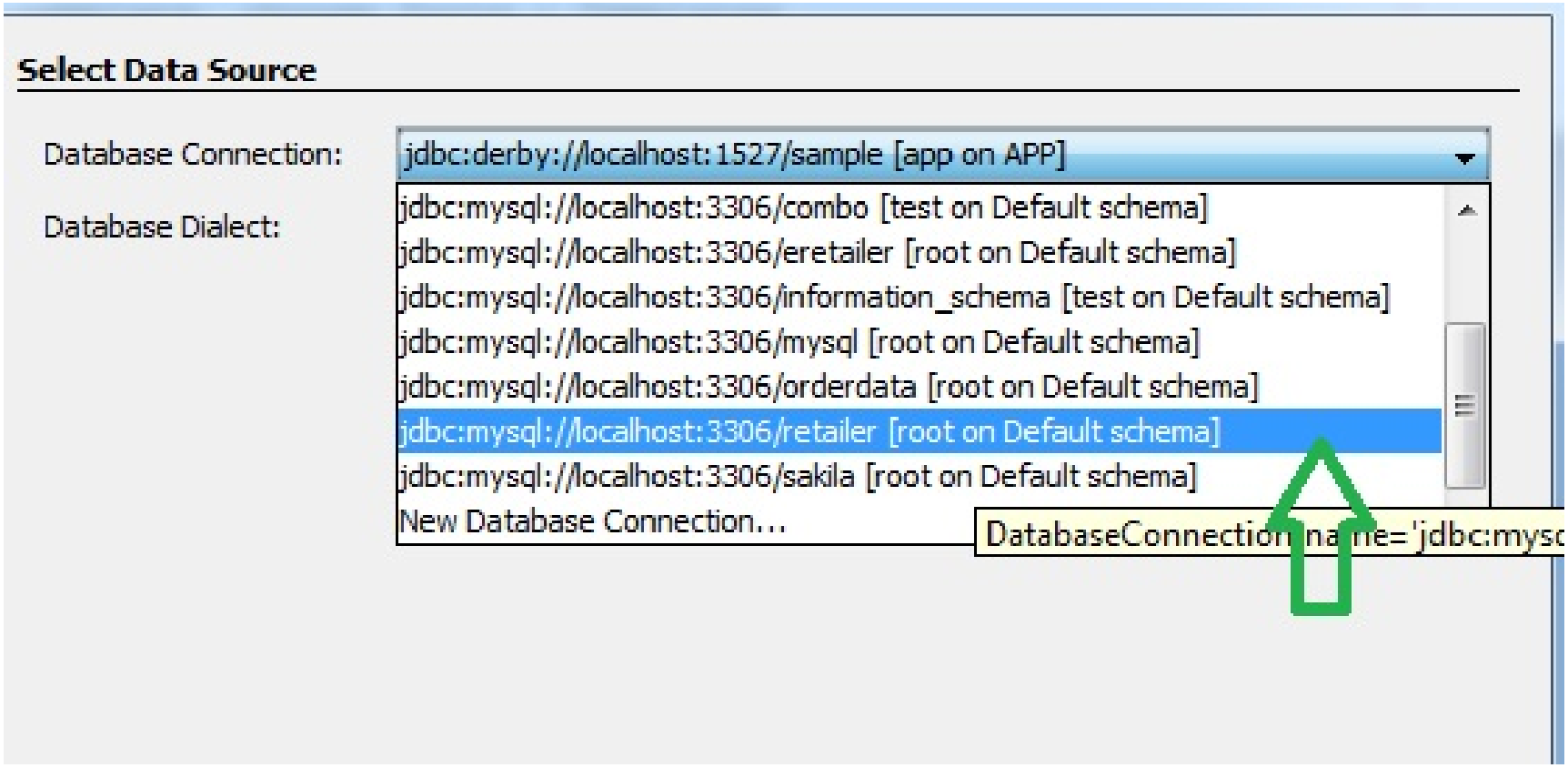
Steps to run hibernate example

Step-6: Creating the configuration XML

- Hibernate need a configuration file to create the connection.
- Right click package hibernatetest select New > Other > **Hibernate > Hibernate Configuration Wizard**
- Click Next >
- In next window click the drop down menu of Database Connection and select retailer database connection.

Steps to run hibernate example: Step-6

Step-6: Creating the configuration XML



- Click Finish to create the file.

Steps to run hibernate example

hibernate.cfg.xml

```
1. <hibernate-configuration>
2.   <session-factory>
3.     <property name="hibernate.connection.driver_class">
                                        com.mysql.jdbc.Driver
  </property>
4.     <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/retailer</property>
5.     <property name="hibernate.connection.username">
                                                root</property>
6.     <property name="hibernate.connection.password">
                                                root</property>
7.     <property name="hibernate.connection.pool_size">
                                                  10</property>
8.     <property name="hibernate.dialect">
                  org.hibernate.dialect.MySQLDialect</property>
```

Steps to run hibernate example

```
9. <property name="current_session_context_class">
                                        thread</property>
10.<property name="cache.provider_class">
          org.hibernate.cache.NoCacheProvider</property>
11.<property name="show_sql">true</property>
12.<property name="hibernate.hbm2ddl.auto">
                                        update</property>
13.<mapping resource="hibernate.hbm.xml"></mapping>
14.</session-factory>
15.</hibernate-configuration>
```

Steps to run hibernate example

Step-7: Creating the mapping file [hibernate.hbm]

- Mapping file will map relevant java object with relevant database table column.
- Right click project select New > Other > Hibernate **> Hibernate Mapping Wizard**
- click Next name it as hibernate.hbm
- click Next> In next window we have to select Class to Map and Database Table.
- After selecting correct class click OK
  Select Database Table
  Click drop down list and select the table you want to map.
  Code for mapping file.

Steps to run hibernate example: Step-7

```
1. <hibernate-mapping>
2.    <class name="hibernatetest.Customer" table="customers">
3.    <id column="C_ID" name="customerID" type="int">
4.    <generator class="native">
5.    </generator></id>
6.    <property name="customerName">
7.    <column name="name">
8.    </column></property>
9.    <property name="customerAddress">
10.   <column name="address">
11.   </column></property>
12.   <property name="customerEmail">
13.   <column name="email">
14.   </column></property>
15.   </class></hibernate-mapping>
```

Steps to run hibernate example: Step-7

Step-7: Creating the mapping file [hibernate.hbm]

- property name = variable name of the POJO class
- column name = database column that maps with previous variable

Steps to run hibernate example

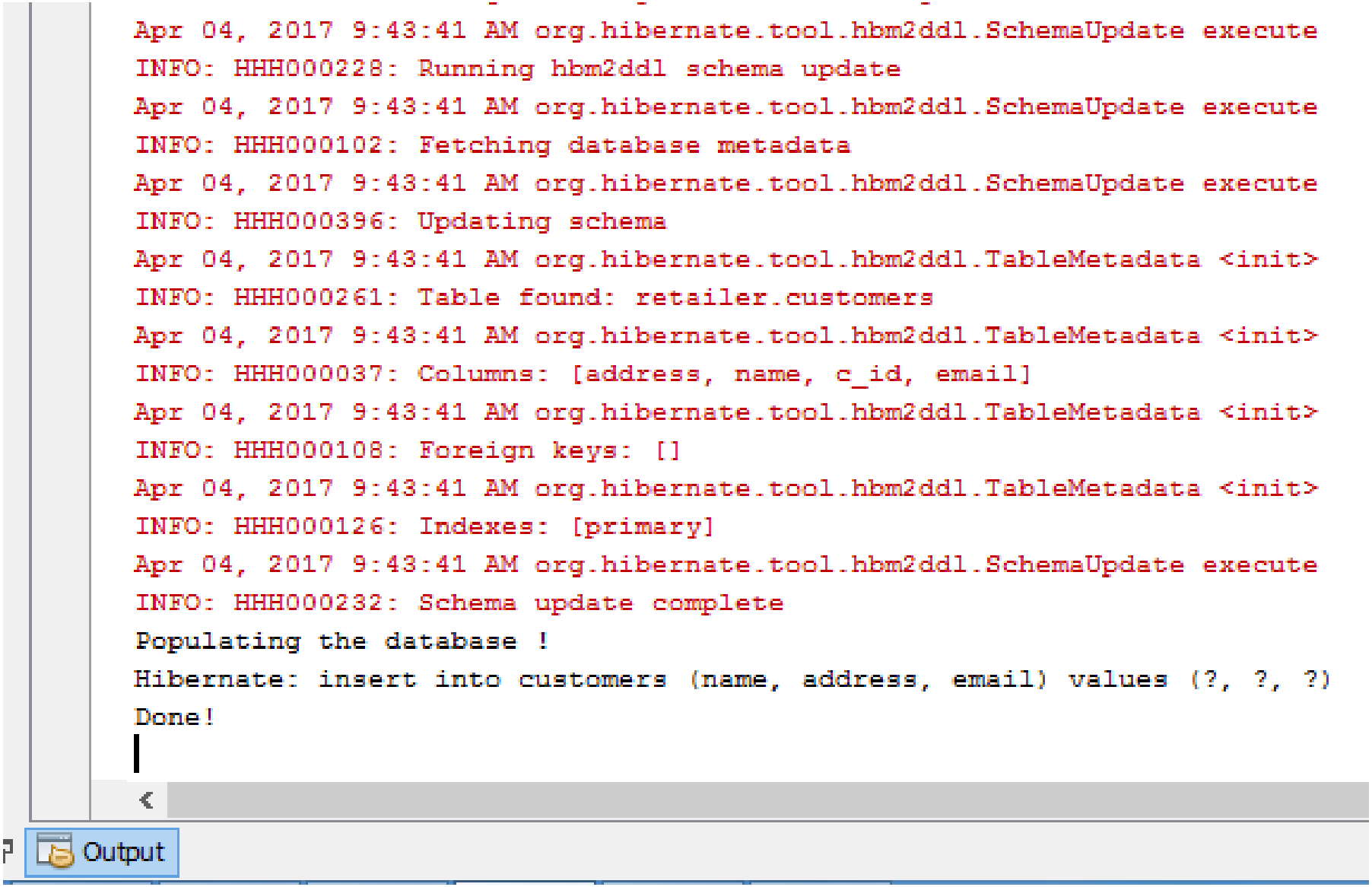
Step-8: Now java program to insert record into the database

```
1. package hibernatetest;
2. import org.hibernate.Session;
3. import org.hibernate.SessionFactory;
4. public class HibernateTest {
5.       public static void main(String[] args) {
6.           Session session = null;
7. try
8. {
9. SessionFactory sessionFactory = new
   org.hibernate.cfg.Configuration().configure().buildSessi
   onFactory();
```

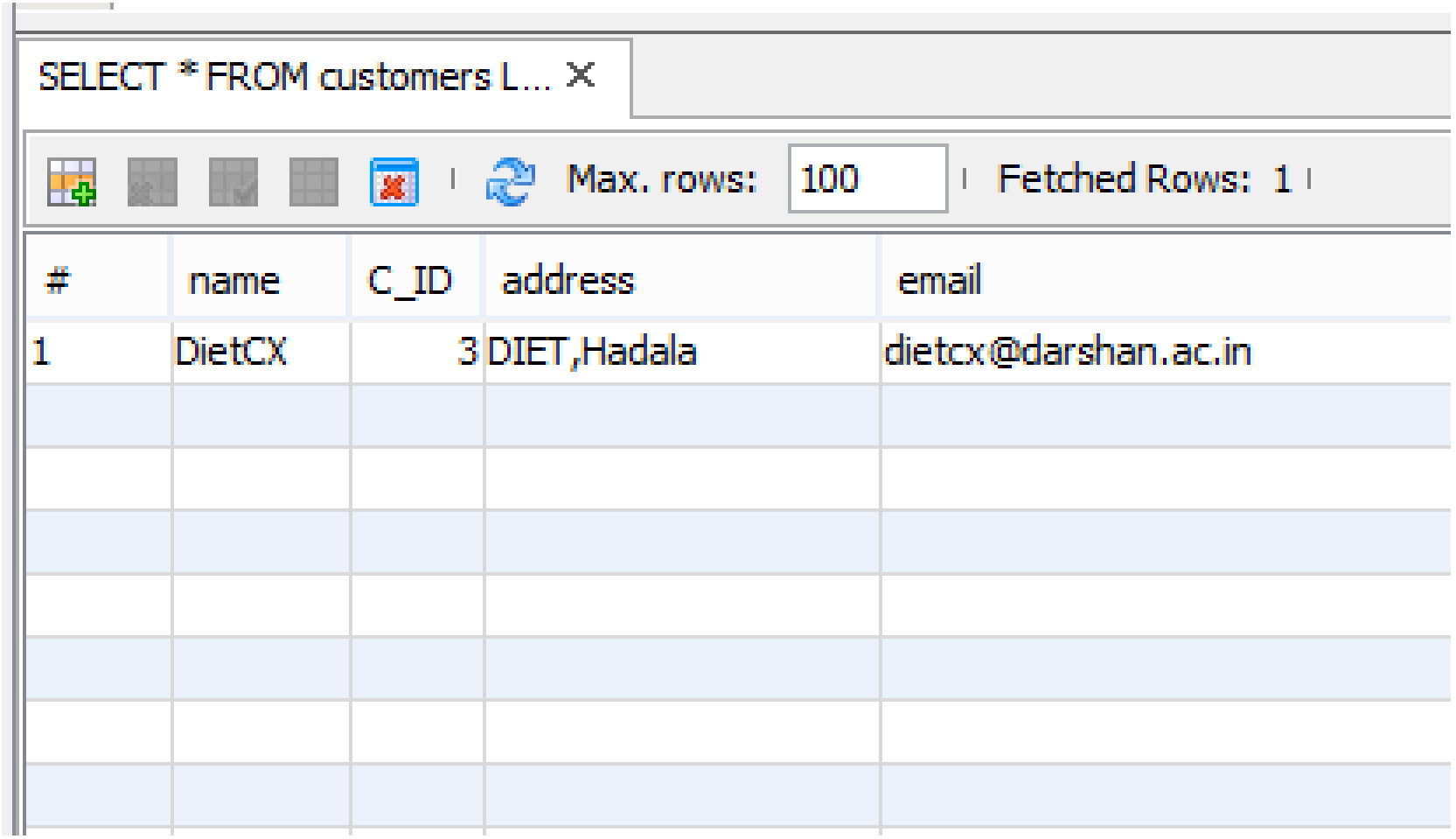
Steps to run hibernate example

```
10.session =sessionFactory.openSession();
11.         session.beginTransaction();
12.         System.out.println("Populating the database !");
13.         Customer customer = new Customer();
14.         customer.setCustomerName("DietCX");
15.         customer.setCustomerAddress("DIET,Hadala");
16.         customer.setCustomerEmail("dietcx@darshan.ac.in");
17.         session.save(customer);
18.         session.getTransaction().commit();
19.         System.out.println("Done!");
20.         session.flush();
21.         session.close();
22.       }catch(Exception e)
       {System.out.println(e.getMessage());     } } }
```

Steps to run hibernate example:output



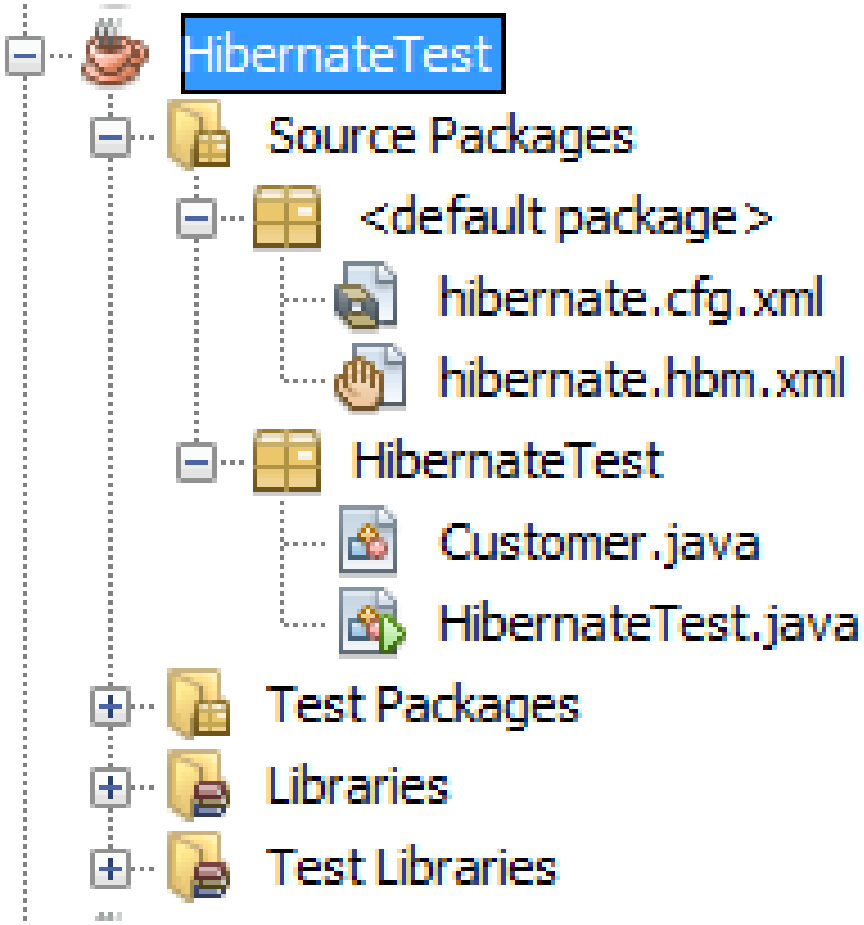
Steps to run hibernate example: output


SELECT * FROM customers L...

Max. rows: 100 Fetched Rows: 1

#	name	C_ID	address	email
1	DietCX	3	DIET,Hadala	dietcx@darshan.ac.in

Hibernate Program Hierarchy



Hibernate with Annotation

- The hibernate application can be created with annotation.
- There are many annotations that can be used to create hibernate application such as @Entity, @Id, @Table etc.
- Hibernate Annotations are based on the JPA 2 specification and supports all the features.
- All the JPA annotations are defined in the javax.persistence.* package.
- Hibernate **EntityManager** implements the interfaces and life cycle defined by the JPA specification.

Hibernate with Annotation

Advantage

- The core advantage of using hibernate annotation is that you don't need to create mapping (hbm) file. Here, hibernate annotations are used to provide the meta data.

GTU Questions

1	Explain the Hibernate cache architecture.
2	What is HQL? How does it different from SQL? List its advantages.
3	What is OR mapping? Give an example of Hibernate XML mapping file.
4	What is HQL? How does it differ from SQL? Give its advantages.
5	Draw and explain the architecture of Hibernate.
6	Explain architecture of Hibernate.
7	Explain architecture of Spring MVC Framework. Explain all modules in brief.
8	What is O/R Mapping? How it is implemented using Hibernate. Explain with example.
9	What are the advantages of Hibernate over JDBC?
10	What is hibernate? List the advantages of hibernate over JDBC.
11	Develop program to get all students data from database using hibernate. Write necessary xml files.