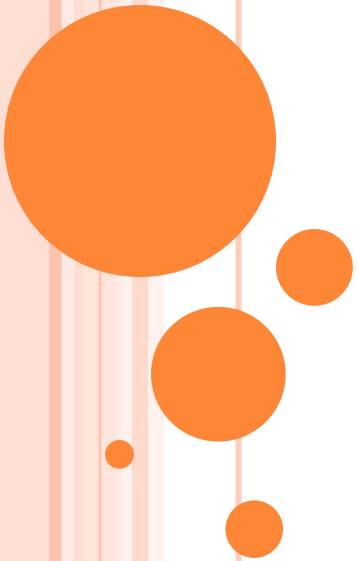
Chap:11

Hibernate



Syllabus Content

- Introduction to Hibernate
- Need for hibernate
- Features of hibernate
- Disadvantages of Hibernate
- Exploring Hibernate Architecture
- Jars files of hibernate.
- Hibernate Configuration file
- Hibernate Mapping file
- Hibernate Annotation
- Hibernate Inheritance
- Hibernate Sessions

- **Introduction to Hibernate**

Introduction to Hibernate

- It was created by Gavin King in 2001.

- Hibernate is an open-source ORM(Object-Relational Mapping) solution for Java applications.

- Hibernate raised as an open source persistent framework.
- It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

- Hibernate provides data query and retrieval facilities that significantly reduce development time.

- **Needs of Hibernate**

Needs of Hibernate

- with JDBC, mapping between **Java** objects and database tables is create manually.
- **Hibernate** reduces lines of code by maintaining object-table mapping itself and returns result to application in form of **Java** objects.

- **Hibernate**, with Transparent Persistence, cache is set to application work space.
- **Hibernate ORM(object relational mapping) (Hibernate** in short) is an tool for the**Java programming** language.
- It provides a framework for mapping an object-oriented domain model to a relational database.
- It generates SQL calls and relieves the developer from the manual handling and object conversion of the result set.

- **Features of Hibernate**

Features (Advantages) of Hibernate

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.

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.

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.

Features of Hibernate

- 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.

- **Disadvantages of Hibernate**

Disadvantages of Hibernate

1.Lots of API to learn: A lot of effort is required to learn Hibernate. So, not very easy to learn hibernate easily.

2.Debugging: Sometimes debugging and performance tuning becomes difficult.

3.Slower than JDBC: Hibernate is slower than pure JDBC as it is generating lots of SQL statements in runtime.

4.Not suitable for Batch processing: It advisable to use pure JDBC for batch processing.

- **Exploring Hibernate Architecture**

Exploring Hibernate Architecture

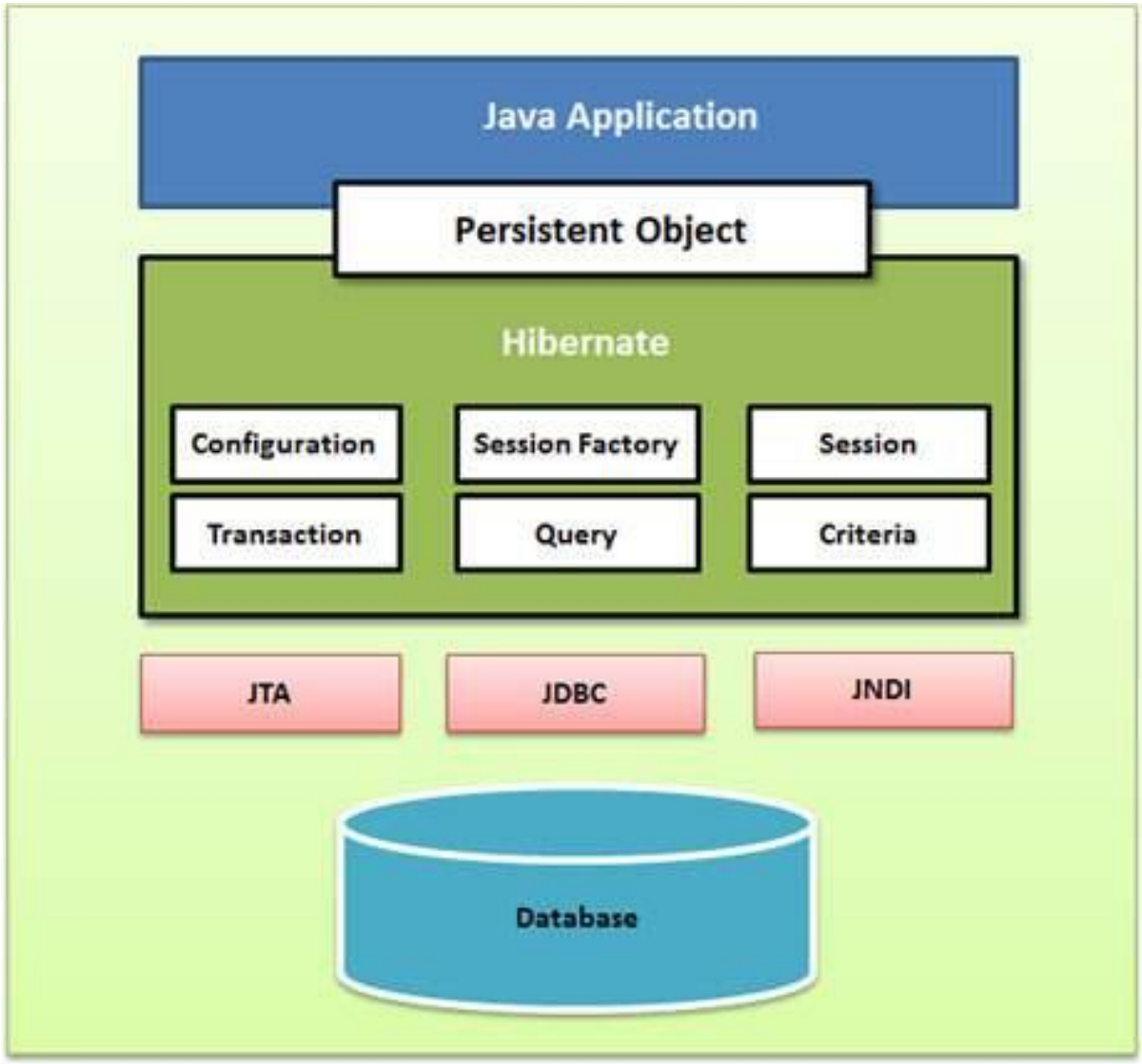
- The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

- The Hibernate architecture is categorized in four layers.

- 1.Java application layer
- 2.Hibernate framework layer
- 3.Backhand api layer
- 4.Database layer

- Let's see the diagram of hibernate architecture:

Exploring Hibernate Architecture



Exploring Hibernate Architecture

- Hibernate uses various existing Java APIs, like JDBC, Java Transaction API(JTA), and Java Naming and Directory Interface (JNDI).

- JDBC provides a runtime level of abstraction of functionality common to relational databases, allowing almost any database with a JDBC driver to be supported by Hibernate.

- JNDI and JTA allow Hibernate to be integrated with J2EE application servers.

- Following section gives brief description of each of the class objects involved in Hibernate Application Architecture Like….
- 1.configuration
- 2.session factory
- 3.session
- 4.Transaction
- 5.Query
- 6.Criteria

1.Configuration

- The Configuration object is the first Hibernate object you create in any Hibernate application and usually created only once during application initialization.

- It represents a configuration or properties file required by the Hibernate.

- **The Configuration object provides two keys components:**
  - **Database Connection**: This is handled through one or more configuration files supported by Hibernate. These files are hibernate. Properties and hibernate.web.xml.
  - **Class Mapping Setup** :- This component creates the connection between the Java classes and database tables..

2.SessionFactory

- The Session Factory is a factory of session and client of Connection Provider.
- It holds second level cache of data.
- The org.hibernate.SessionFactory interface provides factory method to get the object of Session.
- The Session Factory is heavyweight object so usually it is created during application start up and kept for later use.
- You would need one Session Factory object per database using a separate configuration file.
- So if you are using multiple databases then you would have to create multiple Session Factory objects.

3.Session

- A Session is used to get a physical connection with a database.
- The session object provides an interface between the application and data stored in the database.
- The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.
- Persistent objects are saved and retrieved through a Session object.
- 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 them as needed.

4.Transaction

- The transaction object specifies the atomic unit of work with the database and most of the RDBMS supports transaction functionality.
- It is optional.
- Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

5.Query

- 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.

6.Critearia

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

- **Downloading and Configuring and necessary files**

▪Downloading and Configuring and necessary files

we need to follow following steps:

- Create the Persistent class
- Create the mapping file for Persistent class
- Create the Configuration file
- Create the class that retrieves or stores the persistent object

Hibernate Configuration file

Hibernate Configuration file

- the mapping information that defines how your Java classes relate to the database tables.
- Hibernate also requires a set of configuration settings related to database and other related parameters.

- All such information is usually supplied as a standard Java properties file called **hibernate.properties**, or as an XML file named **hibernate.cfg.xml**.

- **Hibernate Properties:**
- **1. hibernate.dialect:**
- This property makes Hibernate generate the appropriate SQL for the chosen database.
- **2. hibernate.connection.driver_class:**
- The JDBC driver class.

3.hibernate.connection.url
The JDBC URL to the database instance.
4. **hibernate.connection.username**
The database username.
5.hibernate.connection.password
The database password.

- **Hibernate Mapping file**

Hibernate Mapping file

- An Object/relational mappings are usually defined in an XML document.

- This mapping file instructs Hibernate — how to map the defined class or classes to the database tables?

- Though many Hibernate users choose to write the XML by hand, but a number of tools exist to generate the mapping document. These include **XDoclet, Middlegen** for the advanced Hibernate users.

- Let's try to understand the simple hibernate example:

Hibernate Mapping file

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name = "Employee" table = "EMPLOYEE">

      <meta attribute = "class-description">
         This class contains the employee detail.
      </meta>

      <id name = "id" type = "int" column = "id">
         <generator class="native"/>
      </id>

      <property name = "firstName" column = "first_name" type = "string"/>
      <property name = "lastName" column = "last_name" type = "string"/>
      <property name = "salary" column = "salary" type = "int"/>

   </class>
</hibernate-mapping>
```

Hibernate Mapping file

- The mapping document is an XML document having **<hibernate-mapping>** as the root element, which contains all the **<class>**elements.

- The **<class>** elements are used to define specific mappings from a Java classes to the database tables.

- The **<meta>** element is optional element and can be used to create the class description.

- The **<id>** element maps the unique ID attribute in class to the primary key of the database table.

- **Hibernate Annotation**

Hibernate Annotation

- Hibernate Annotations is the powerful way to provide the metadata for the Object and Relational Table mapping.

- All the metadata is clubbed into the POJO java file along with the code, this helps the user to understand the table structure and POJO simultaneously during the development.

- If you going to make your application portable to other EJB 3 compliant ORM (object research management)applications, you must use annotations to represent the mapping information, but still if you want greater flexibility, then you should go with XML-based mappings.

Environment Setup for Hibernate Annotation:

- First of all you would have to make sure that you are using JDK 5.0 otherwise you need to upgrade your JDK to JDK 5.0 to take advantage of the native support for annotations.

Hibernate Annotation

- Second, you will need to install the Hibernate 3.x annotations distribution package, available from the source forge: (Download Hibernate Annotation) and copy **hibernate-annotations.jar, lib/hibernate-comons-annotations.jar** and **lib/ejb3-persistence.jar** from the Hibernate Annotations distribution to your CLASSPATH.

- **Hibernate Inheritance**

Hibernate Inheritance

- We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

- Table Per Hierarchy

- Table Per Concrete class

- Table Per Subclass

- **1. Table Per Hierarchy:**

- In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But null values are stored in the table .

- **2. Table Per Concrete class:**

- In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

- **3. Table Per Subclass**

- In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

- **Hibernate Sessions**

Hibernate Sessions

- 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.

Assignment Question

Q.1 what is hibernate ? Explain uses of hibernate.

Q.2 Explain features of hibernate

Q.3 Write disadvantages of hibernate.

Q.4 Explain architecture of hibernate.

Q.5 Explain Hibernate Configuration file

Q.6 Explain Hibernate Mapping file

Q.7 Explain Hibernate Annotation

Q.8 Hibernate Inheritance

Q.9 Hibernate Sessions