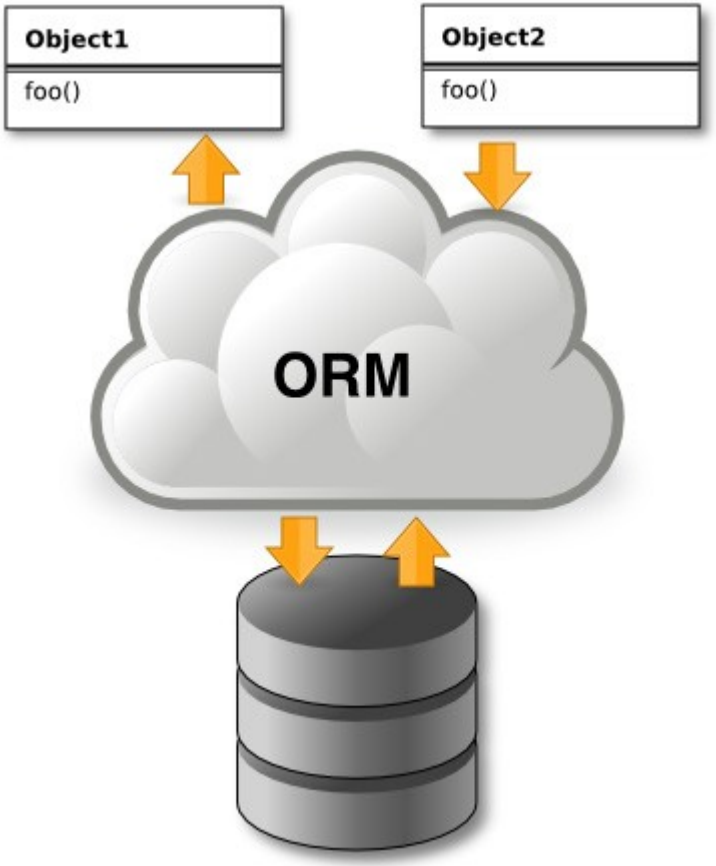
Hibernate

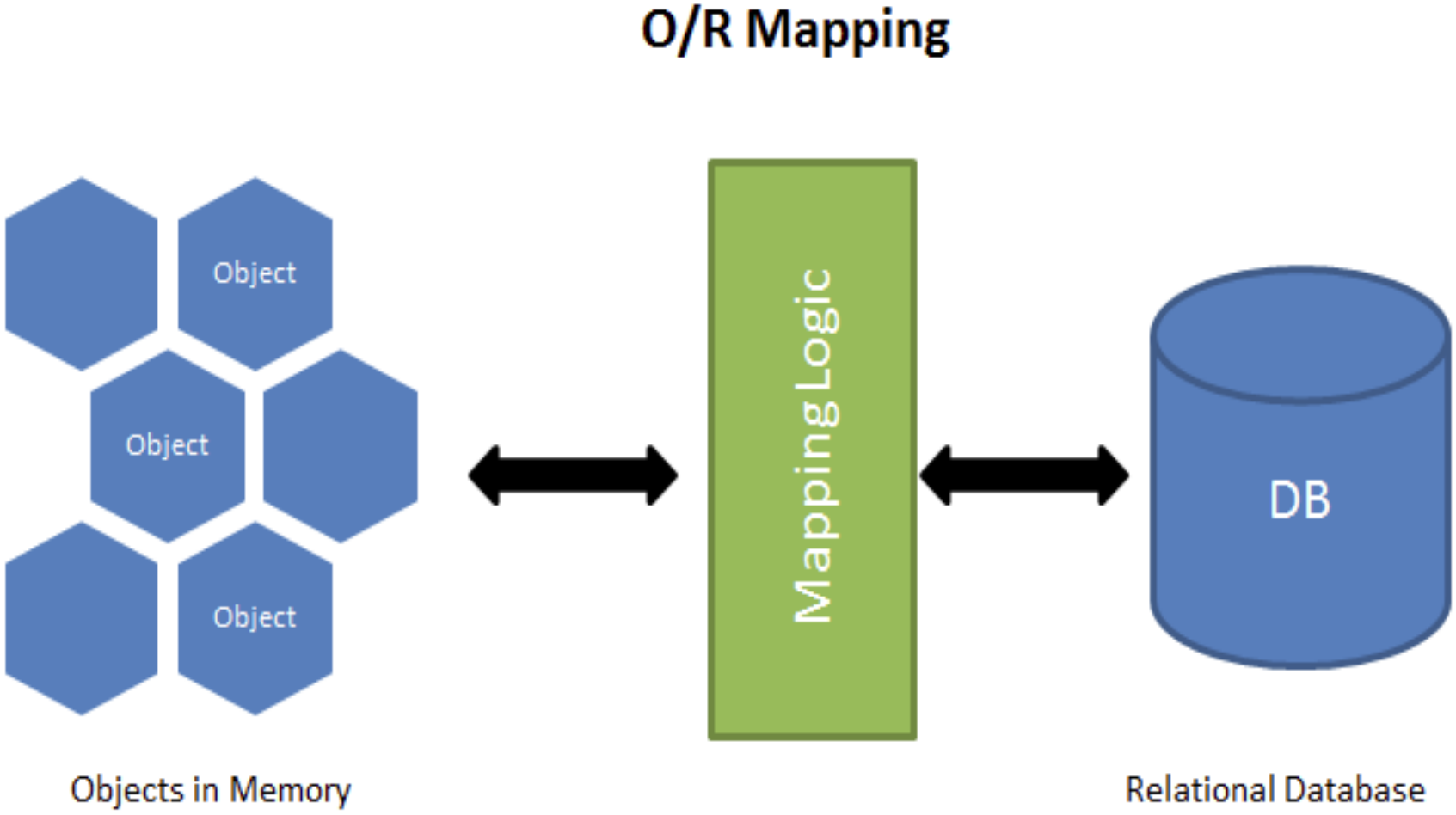
Exploring object-relational mapping

Team Emertxe



ORM

What is ORM



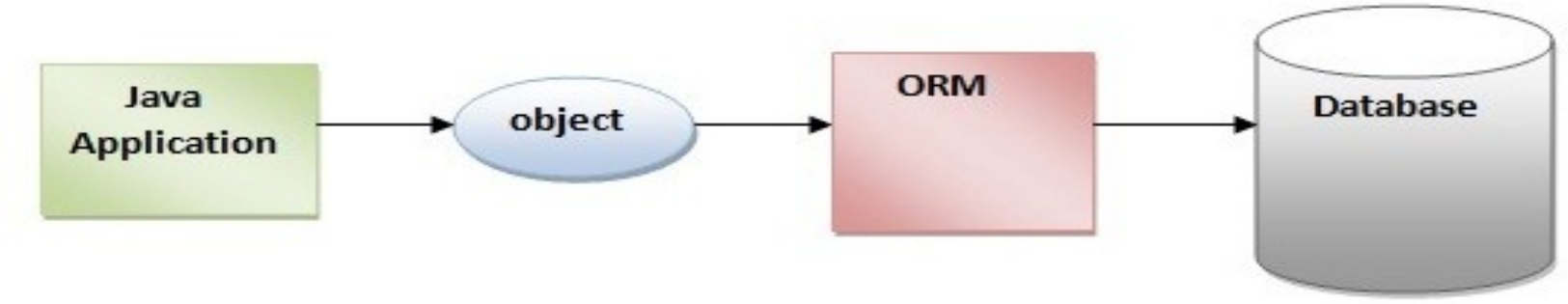
EMERTXE

What is ORM

The ORM tool stands for Object Relational Mapping tool and it takes care of generating the sql statements.

It executes the sql statement and finally processing the result. It also handles the exceptions occurred in the program.

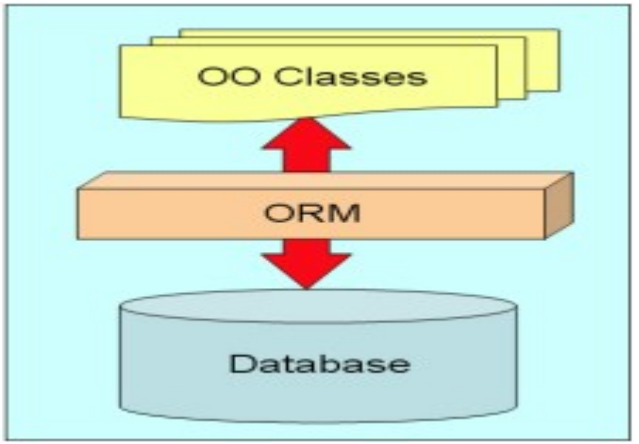
Example of ORM tools: Hibernate, TopLink, JPA, iBatis



EMERTXE

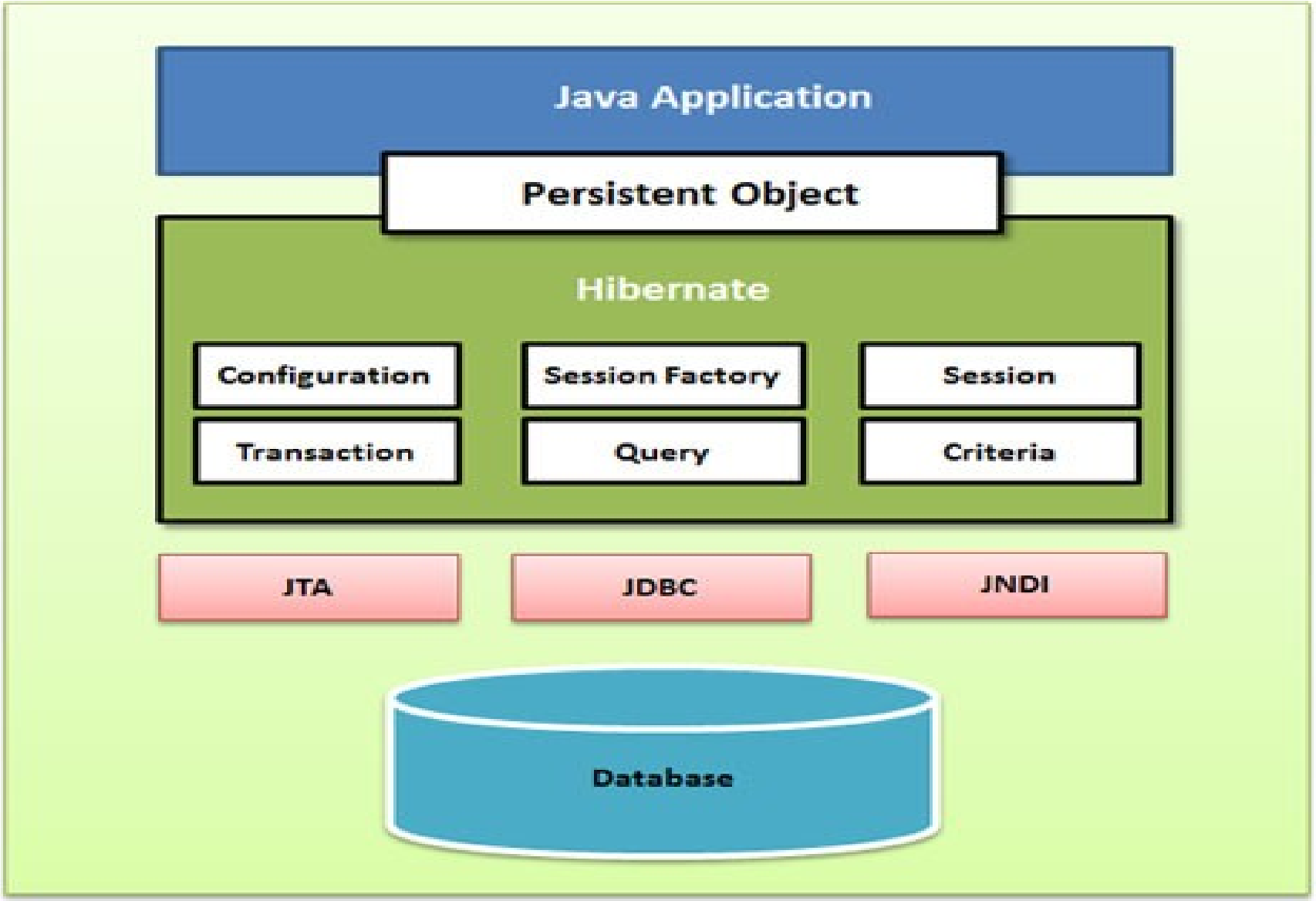
Features of ORM

- ORM tool support relationship amongst the objects.
- It supports lazy loading and many other optimization functions.
- It also supports the concurrency.
- It also supports primary and secondary level cache.
- Transaction management also a very important feature of the database. The ORM tools also supports transaction management and error handling.



Hibernate

What is Hibernate



What is Hibernate

- Hibernate is feature rich tool which is used to develop enterprise applications.
- It can be used with Java SE application and web applications.
- It can easily used with JSP, Servlets, Struts. Spring Web, JSF and other Java based we application development frameworks.
- Hibernate is also used with JPA as persistence provider.
- Hibernate also supports primary-level and secondary-level caching. You can use any cache provider library.

Hibernate

- 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. It is a programming technique that maps the object to the data stored in the database.

Advantages of Hibernate

- Opensource and Lightweight: Hibernate framework is opensource and lightweight.
- Fast performance: The performance of hibernate framework is fast because cache is internally used in hibernate framework.
- Database Independent query: HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries.
- 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.

Elements of Hibernate

SessionFactory

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

- **Session**

  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

Elements of hibernate

- **Transaction**

  The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

- **ConnectionProvider**

  It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

- **TransactionFactory**

  It is a factory of Transaction. It is optional.

Steps To Create Hibernate Applications

Steps To Create Hibernate Application

Step 1: Create database and tables

Step 2: Create POJO class

Step 3: Create mapping meta-data

Step 4: Write code to perform create, read, update and delete operations.

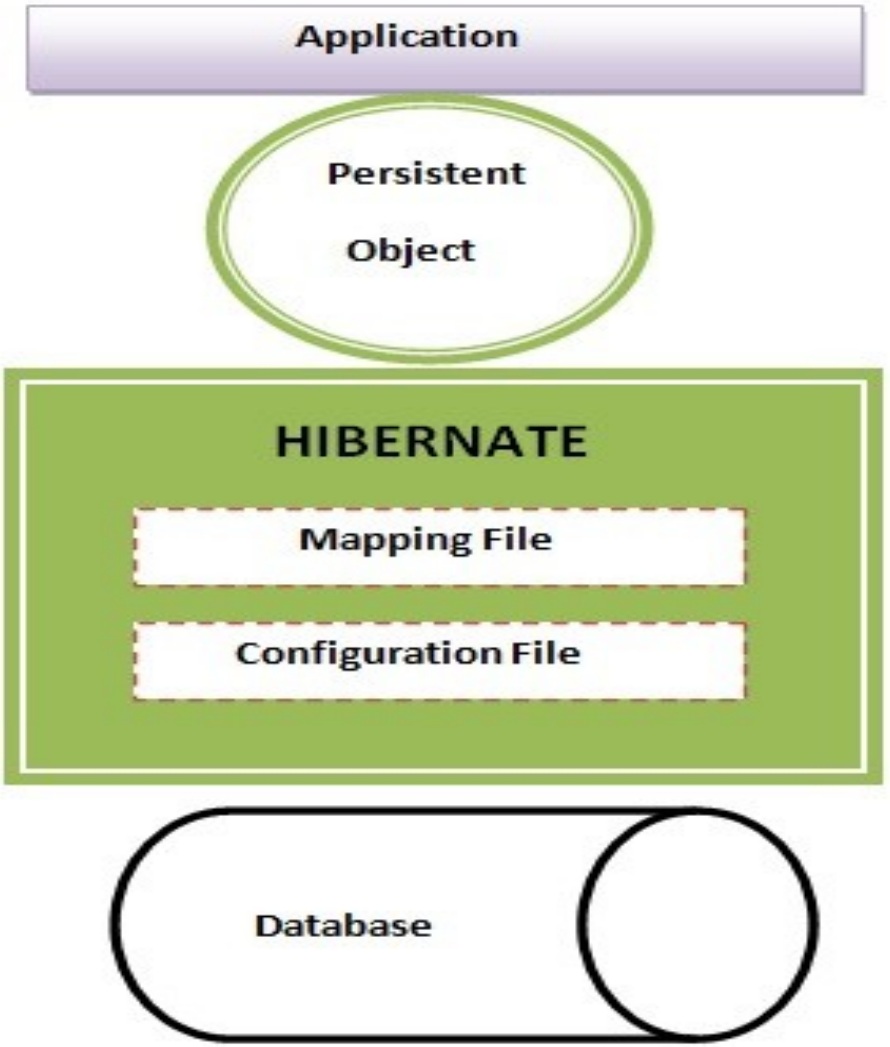
Component Of Hibernate Application

Component of Hibernate Application

- Hibernate.cfg.xml
  - Hibernate configuration files
- POJO Class
  - Plain Old Java Objects
- Hibenate.hbm.xml
  - Hibernate mapping file
- Client Class
  - The Java file

The Configuration File/ Hibernate.cfg.xml

Hibernate Configuration File



Hibernate Configuration XML File

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
     <!-- in the next slide-->
    </session-factory>
</hibernate-configuration>
```

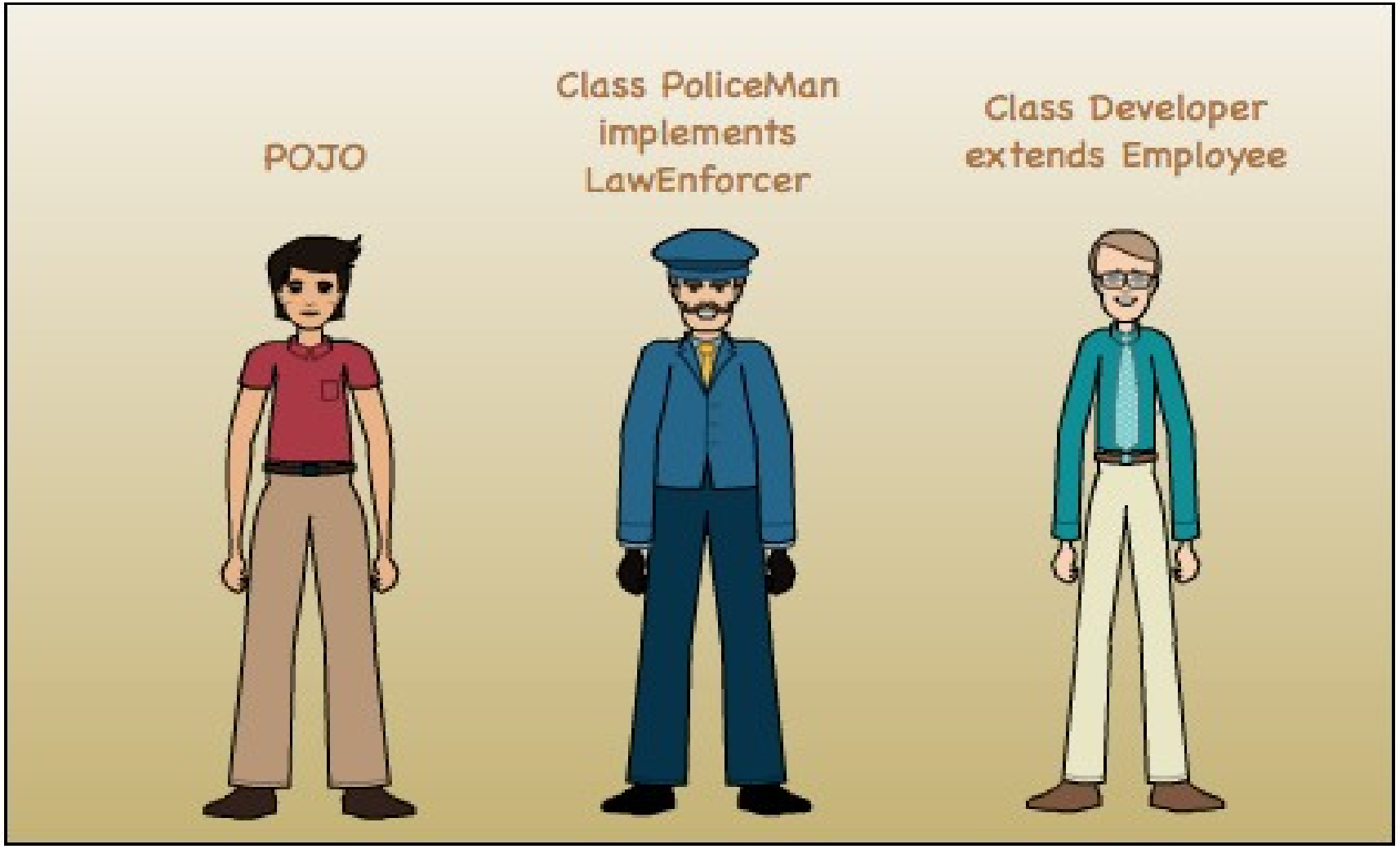
EMERTXE

Hibernate Configuration XML File

```
<session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">
         org.hibernate.dialect.Oracle9Dialect
        </property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">oracle</property>
        <property name="connection.driver_class">
        oracle.jdbc.driver.OracleDriver</property>
       <mapping resource="employee.hbm.xml"/>
</session-factory>
```

The POJO Class

Hibernate POJO Class



POJO class

- Plain old Java object (POJO) is an ordinary Java object, not bound by any special restriction and not requiring any class path.

- **Persistent Objects** are generally those objects that exist in memory even beyond the duration of the process that creates it. These objects are then stored in the database.

Hibernate POJO Class

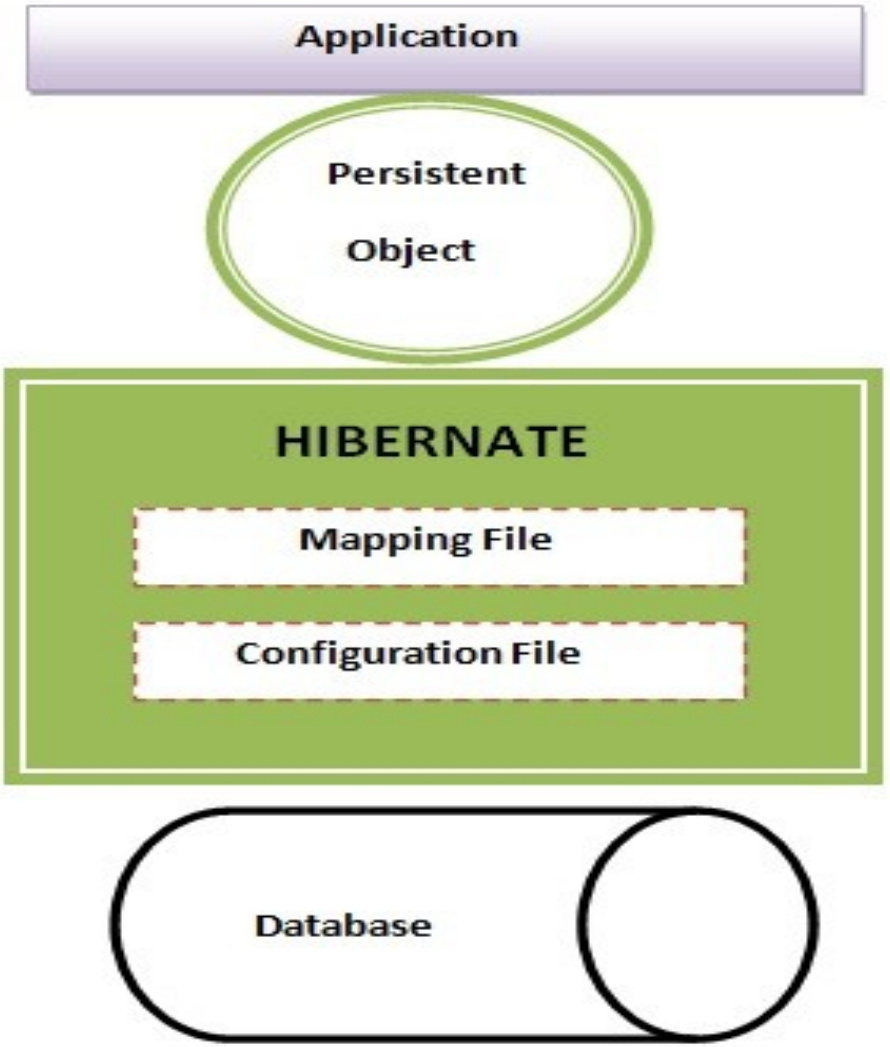
```
public class Employee {
private int id;
private String firstName,lastName;
public int getId() {
 return id;
}
public String getFirstName() {  return firstName;
}
public String getLastName() {  return lastName;
}
```

Hibernate POJO Class

```
public void setId(int id) {
this.id = id;
}
public void setFirstName(String firstName) {
this.firstName = firstName;
}
public void setLastName(String lastName) {
this.lastName = lastName;
}
```

The MappingFile/ Hibernate.hbm.xml

Hibernate Mapping File



Hibernate Mapping XML File

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
 <class name="com.Employee"
  table="EMPLOYEE">
  <!-- in next slide-->
  </class>
  </hibernate-mapping>
```

Elements of Mapping File

- **hibernate-mapping** is the root element in the mapping file.
- **class** - It specifies the Persistent class.
- **id** - It specifies the primary key attribute in the class.
- **generator** -It is the subelement of id. It is used to generate the primary key.
- **property** -It is the subelement of class that specifies the property name of the Persistent class.

Generator class

assigned

- It is the default generator strategy if there is no <generator> element .

increment

- It generates the unique id only if no other process is inserting data into this table

native

- It uses identity, sequence depending on the database vendor

Hibernate Dialects class

- For connecting any hibernate application with the database, you must specify the SQL dialects. There are many Dialects classes defined for RDBMS in the org.hibernate.dialect package.

Log4j

- Logging enables the programmer to write the log details into a file permanently.

There are two ways to perform logging using log4j

- By log4j.xml file (or)
- By log4j.properties file

There are two ways to perform logging using log4j using xml file:

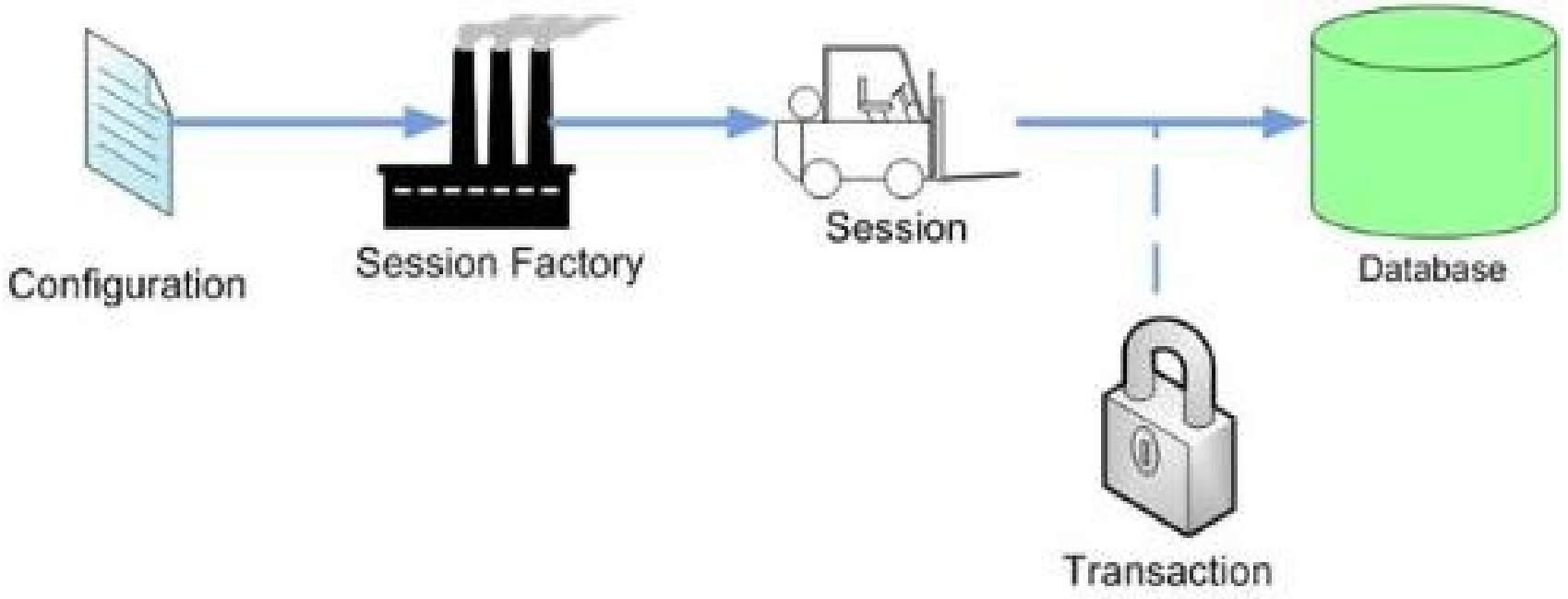
- Load the log4j jar files with hibernate
- Create the log4j.xml file inside the src folder (parallel with hibernate.cfg.xml file)

Hibernate Mapping XML File

```
<class name="com.Employee" table="EMPLOYEE">
<id name="id">
<generator class="assigned"></generator>
</id>
<property name="firstName"></property>
<property name="lastName"></property>
</class>
```

The Client Class/ Java Class

Component Inside the Client Class



The Client Code

```
Configuration cfg=new Configuration();
cfg.configure("hibernate.cfg.xml");
SessionFactory factory=cfg.buildSessionFactory();
Session session=factory.openSession();
Transaction t=session.beginTransaction();
```

The Client Code

```
Employee e1=new Employee();
e1.setId(115);
e1.setFirstName("Gavin");
e1.setLastName("King");
session.persist(e1);//persisting the object
t.commit();//transaction is commited
session.close();
```

Hibernate Application With Annotation

Steps of Hibernate Application With Annotation

- Create configuration file
- Create POJO class
- Add POJO class in the configuration file
  - <mapping class="com.Employee"/>
- The client class

Hibernate Annotations

- 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.

Persistance class with annotations

- @Entity annotation marks this class as an entity.
- @Table annotation specifies the table name where data of this entity is to be persisted. If you don't use @Table annotation, hibernate will use the class name as the table name by default.
- @Id annotation marks the identifier for this entity.
- @Column annotation specifies the details of the column for this property or field. If @Column annotation is not specified, property name will be used as the column name bydefault.

Hibernate Inheritance Mapping

We can map the inheritance hierarchy classes with the table of the database.

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 nullable values are stored in the table .

- Table per hierarchy using xml file.
- Table per hierarchy using Annotation file

Table per hierarchy using Annotation file

```
@Entity
@DiscriminatorValue("prod")
public class Sale extends Product{

@Column(name="Prod_nm")
private String  Prod_nm;

@Column(name="Price")
private float price;

//setters and getters
}
```

Collection mapping

The Hibernate mapping element used for mapping a collection depends upon the type of interface. There are many subelements of <class> elements to map the collection. They are <list>, <bag>, <set> and <map>.

The bag is just like List but it doesn't require index element.

List

There are three subelements used in the list:

- <key> element is used to define the foreign key in this table based on the Question class identifier.
- <index> element is used to identify the type. List and Map are indexed collection.
- <element> is used to define the element of the collection.

Collection Elements

The collection elements can have value or entity reference (another class object). We can use one of the 4 elements

- element
- component-element
- one-to-many, or
- Many-to-many

The element and component-element are used for normal value such as string, int etc. whereas one-to-many and many-to-many are used to map entity reference.

Mapping file

```
<class name="com.Question" table="q100">
        <id name="id">
         <generator class="increment"></generator>
         </id>
          <property name="qname"></property>

         <list name="answers" table="ans100">
         <key column="qid"></key>
         <index column="type"></index>
          <element column="answer" type="string"></element>
          </list>

         </class>
```

EMERTXE

Mapping Set

```
<class name="MappingList.Question" table="ques">
    ...
    <set name="answers" table="ans">
        <key column="qid"></key>
        <element column="answer"
type="string"></element>
        </set>
        </class>
```

Lazy collection

Lazy loading has its own advantages, it is not loading lots of objects but only when you need them.

Hibernate now can "lazy-load" the children, which means that it does not actually load all the children when loading the parent. Instead, it loads them when requested to do so.Lazy-loading can help improve the performance significantly since often you won't need the children and so they will not be loaded.

Lazy and Eager Collection

- EAGER: Convenient, but slow
- LAZY: More coding, but much more efficient

Caching in Hibernate

Hibernate caching improves the performance of the application by pooling the object in the cache.

There are mainly two types of caching:

- **First Level Cache** :-Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.
- **Second level cache** :- SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

HQL

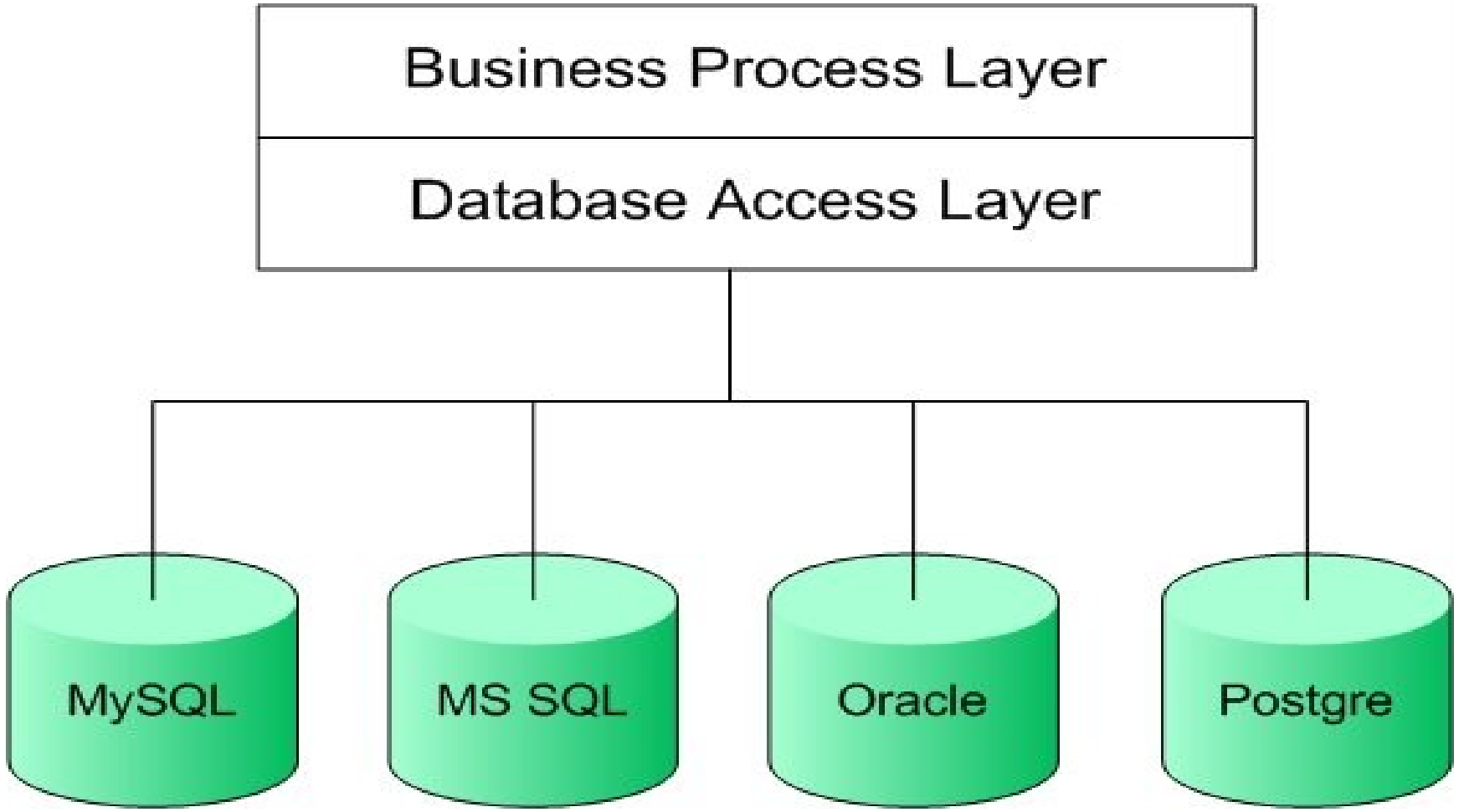
Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL. So it is database independent query language.

Web Application with Hibernate

Steps to Create Web Application Using Hibernate

- Creating a JSP page (index.jsp)
- Creating a Servlet (MyServlet.java)
- Creating a POJO class (User.java)
- Creating a mapping file (Hiberntae.hbm.xml)
- Creating a DAO class (UserDAO.java)
- Creating a configuration file (Hibernate.cfg.xml)

DAO

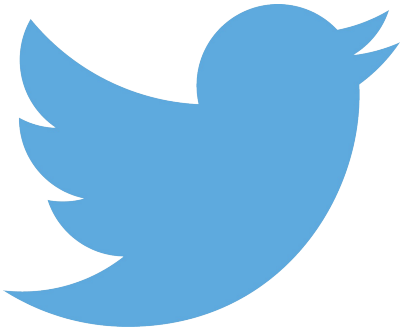

Business Process Layer
Database Access Layer
MySQL
MS SQL
Oracle
Postgre

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You