**CREATING THE DAO**

1. DAO (Data Access Object) is a class , which is used to ***access data*** from database.

2. All database related operations required by an application are defined as methods in a DAO class. For example our DAO will contain methods for **adding employee record to the database** , **searching an employee in the database** , **updating employee record** etc

3. Like POJO , it is also created on per table basis.

4. In our case , since we have only 1 table , so we would have 1 DAO class called **EmployeeDAO** in the package **empmgmt.dao**

5. Following will be the total methods in our **EmployeeDAO** class:

a. **addEmployee( )**

b. **findEmployeeById( )**

c. **getAllEmployees( )**

d. **updateEmployee( )**

e. **deleteEmployee( )**

**CREATING THE addEmp( ) METHOD**

1. The **addEmp( )** method of our **EmpDAO** class will add a new record in the **Emp** table of the database.

2. Following are it's important points:

**a. It will accept an EmpPojo object as argument containing all the fields of data**

**b. It will get a Connection object from DBConnection class using the method getConnection( )**

**c. It will then create a PreparedStatement object with insert query , fill all the values in it and add the record to the Emp table**

**d. If insertion is successful it will return true otherwise return false.**

**e. It will not handle any SQLException and will simply pass it on to it's caller**

3. Based upon the above facts the prototype of the method **addEmployee( )** is:

***public static boolean addEmp(EmpPojo obj) throws SQLException***

4. Following is it's code:

***public class EmpDAO {***

***public static boolean addEmp(EmpPojo e)throws SQLException{***

***Connection conn=DBConnection.getConnection();***

***PreparedStatement ps=conn.prepareStatement("Insert into emp values(?,?,?)");***

***ps.setInt(1, e.getEmpno());***

***ps.setString(2, e.getEname());***

***ps.setDouble(3,e.getSal());***

***int x=ps.executeUpdate();***

***return x==1;***

***}***

**CREATING THE getAllEmp( ) METHOD**

1. The **getAllEmp( )** method of our **EmpDAO** class will return all the records from the **Emp** table .

2. Following are it's important points:

**a. It will accept no argument**

**b. It will get a Connection object from DBConnection class using the method getConnection( )**

**c. It will then create a Statement object with SELECT and execute it**

**d. It will receive the ResultSet**

**e. Then it will traverse the ResultSet , fetch one row at a time , store it in EmpPojo object and add this object to the ArrayList.**

**e. Finally it will return this ArrayList**

**f. It will not handle any SQLException and will simply pass it on to it's caller**

3. Based upon the above facts the prototype of the method **getAllEmp( )** is:

***public static ArrayList<EmpPojo> getAllEmp() throws SQLException***

4. Following is it's code:

***public static ArrayList<EmpPojo> getAllEmp() throws SQLException***

***{***

***Connection conn=DBConnection.getConnection();***

***Statement st=conn.createStatement();***

***ResultSet rs=st.executeQuery("Select \* from emp");***

***ArrayList<EmpPojo> empList=new ArrayList<>();***

***while(rs.next())***

***{***

***int eno=rs.getInt(1);***

***String ename=rs.getString(2);***

***double sal=rs.getDouble(3);***

***EmpPojo obj=new EmpPojo(eno,ename,sal);***

***empList.add(obj);***

***}***

***return empList;***

***}***

**CREATING THE findEmpById( ) METHOD**

1. The **findEmpById( )** method of our **EmpDAO** class will search the **Emp** table for a given **Empno** and return the record(**Empno** , **Ename** and **Sal**) of that Employee

2. Following are it's important points:

**a. It will accept an Employee Number as argument.**

**b. It will get a Connection object from DBConnection class using the method getConnection( )**

**c. It will then create a PreparedStatement object with Select query with where clause, replace the placeholder with Employee Number and execute the query**

**d. If the query returns a record then it will create an EmpPojo object , fill all the column values in it and return it otherwise it will return null.**

**e. It will not handle any SQLException and will simply pass it on to it's caller**

3. Based upon the above facts the prototype of the method **findEmpById( )** is:

***public static EmpPojo findEmpById(int eno) throws SQLException***

4. Following is it's code:

***public static EmpPojo findEmpById(int eno)throws SQLException***

***{***

***Connection conn=DBConnection.getConnection();***

***PreparedStatement ps=conn.prepareStatement("Select \* from emp where eno=?");***

***ps.setInt(1,eno);***

***ResultSet rs=ps.executeQuery();***

***EmpPojo e=null;***

***if(rs.next())***

***{***

***e=new EmpPojo();***

***e.setEname(rs.getString(2));***

***e.setSal(rs.getDouble(3));***

***e.setEmpno(eno);***

***}***

***return e;***

***}***

4. Following is it's code:

***public static EmpPojo findEmpById(int eno)throws SQLException{***

***Connection conn=DBConnection.getConnection();***

***PreparedStatement ps=conn.prepareStatement("Select \* from emp where empno=?");***

***ps.setInt(1, eno);***

***EmpPojo e=null;***

***ResultSet rs=ps.executeQuery();***

***if(rs.next()){***

***e=new EmpPojo();***

***e.setEmpno(rs.getInt(1));***

***e.setEname(rs.getString(2));***

***e.setSal(rs.getDouble(3));***

***}***

***return e;***

***}***