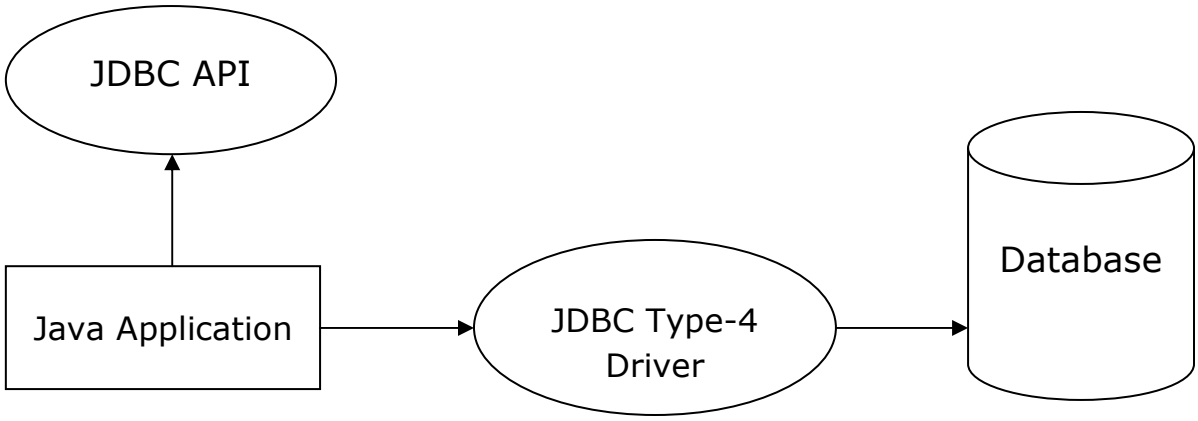
Java JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

1. Type- 1 Driver - JDBC-ODBC Bridge Driver (Removed in JAVA SE8)
2. Type- 2 Driver - Native Driver (Partial Java Driver)
3. Type- 3 Driver - Network Protocol Driver (Full Java Driver)
4. Type- 4 Driver - Thin Driver (Full Java Driver)

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



JDBC (Java Database Connectivity)

The current version of JDBC is based on the X/Open SQL Call Level Interface. The java.sql package contains classes and interfaces for JDBC API.

A list of popular interfaces of JDBC API is given below:

1. Driver interface
2. Connection interface
3. PreparedStatement interface
4. ResultSet interface
5. ResultSetMetaData interface
6. DatabaseMetaData interface

A list of popular classes and interfaces of JDBC API are given below:

1. DriverManager class
2. Connection Class
3. PreparedStatement Class

Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

Which Driver to Use When?

1. Accessing one type of database, such as MySql, Oracle, Sybase, or IBM, the preferred driver type is 4.
2. Java application is accessing multiple types of databases at the same time type 3 is the preferred driver.
3. Type 3 or type 4 driver is not available yet for your database then in this situation Type 2 drivers are useful.
4. The type 1 driver is not considered a deployment-level driver, and is typically used for development and testing purposes only.

Difference between ODBC and JDBC

	ODBC	**JDBC**
1	ODBC Stands for Open Database Connectivity.	JDBC Stands for java database connectivity.
2	Introduced by Microsoft in 1992.	Introduced by SUN Micro Systems in 1997.
3	We can use ODBC for any language like C,C++,Java etc.	We can use JDBC only for Java languages.
4	We can choose ODBC only windows platform.	JDBC is platform independent so we can Use JDBC in any platform.
5	Mostly ODBC Driver developed in native languages like C/C++.	JDBC Stands for java database connectivity.
6	For Java applications it is not recommended to use ODBC because performance will be down due to internal conversion and applications will become platform Dependent.	For Java application it is highly recommended to use JDBC because there we no performance & platform dependent problem.
7	ODBC is procedural.	JDBC is object oriented.

Steps for connectivity between Java program and database

Download the mysql-connector.jar file from the internet.
Move the jar file to lib folder present in the apache-tomcat directory.

Step 1: Creation of Database and Table in MySQL

```
CREATE TABLE Item (ItemID INT(10) Primary Key, ItemName VARCHAR(50));
```

Step 2: Implementation of required Web-pages

Create a form in HTML file, where take all the inputs required to insert data into the database. Specify the Servlet name in it, with the POST method as security is important aspects in database connectivity.

```
<html>
<head> <title> Item Master </title> </head>
<body>
        <form action = "./InsertItem" method = "POST">
        Item ID       :       <input type = "text" name="ItemID"/><br/>
        Item Name   :       <input type = "text" name="ItemName"/><br/>
                        :       <input type = "submit" />
        </form>
</body>
</html>
```

Step 3: Creation of Java Servlet program with JDBC Connection

1. Import all the packages
2. Register the JDBC Driver
3. Open a connection
4. Execute the query, and retrieve the result
5. Clean up the JDBC Environment

Step 4: Create a Servlet that handles user request, performs database operations and finally creates bean object and forwards request to JSP page for display (MVC Architecture)

```
import java.io.IOException;
import java.io.PrintWriter;

import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class InsertItem extends HttpServlet
{
        private Connection getConnection()
```

```
                    throws SQLException, ClassNotFoundException
{
        Connection con = null;
        String dbDriver = "com.mysql.jdbc.Driver";
        String dbURL = "jdbc:mysql:// localhost:3306/";
        String dbName = "Student";
        String dbUser = "root";
        String dbPwd = "root";
        Class.forName(dbDriver);
        con = DriverManager.getConnection(dbURL + dbName,
                                                dbUser,  dbPwd);
        return con;
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException
{
        PrintWriter out = null;
        ItemBean bean = null;
        RequestDispetcher  rd = null

        try
        {
                bean = new ItemBean();

                String query = "INSERT INTO Item VALUES(?, ?, ?, ?)";
                Connection con = this.getConnection();
                String itemID = request.getParameter("ItemID");
                PreparedStatement stmt = con.prepareStatement(query);

                stmt.setInt(1, Integer.valueOf(itemID));
                stmt.setString(2, request.getParameter("ItemName"));

                stmt.executeUpdate();

                query = "SELECT * FROM Item Where ItemID = " +
                stmt = con.prepareStatement(query);
                stmt.executeQuery();

                if (rs.next()) {
                        bean.setItemID(Integer.parseInt(rs.getString("ItemID")));
                        bean.setItemName(rs.getString("ItemName"));
                }

                stmt.close();
                con.close();
                request.Attribute( "itemBean", bean );
```

```
                rd = request.getRequestDispetcher("Display.jsp")
                rd.forward(request, response);
            }
            catch (Exception e) {
                out = response.getWriter();
                out.println("<b>An Error Occurred While Inserting Data </b>");
            }
        }
}
```

FileName: Display.JSP

```
<html>
<head> <title> Display Item </title> </head>
<body>
    <%
        ItemBean bean  = (ItemBean) request.getAttribute("itemBean");
    %>
    <b> Item Inserted Successfully</b>
    <table>
        <tr>
            <td>Item ID</td>
            <td><% = bean.getItemID(); %></td>
        </tr>
        <tr>
            <td>Item ID</td>
            <td><% = bean.getItemName(); %></td>
        </tr>
    </table>
</body>
</html>
```

File : ItemBean

```
public class ItemBean
{
    private int itemID;
    private string itemName;
    public int getItemID () {return itemID}
    public void setItemID (int ID) { itemID = ID}
    public int getItemName () {return itemID}
    public void setItemName (string Name) { itemName = Name}

}
```

Database Access using JSP

Java code can be placed in declaration, expression and scriptlet tags of JSP page and displayed

FileName: ProcessAndDisplay.JSP

```
<html>
<head> <title> Display Item </title> </head>
<body>

<!%
        private Connection getConnection() throws SQLException,
                                                    ClassNotFoundException
        {
                Connection con = null;
                String dbDriver = "com.mysql.jdbc.Driver";
                String dbURL = "jdbc:mysql:// localhost:3306/";
                String dbName = "Student";
                String dbUser = "root";
                String dbPwd = "root";
                Class.forName(dbDriver);

                con = DriverManager.getConnection(dbURL + dbName,
                                                        dbUser,  dbPwd);
                return con;
                }

                public void accessDatabase() throws ServletException, IOException
                {
                PrintWriter out = null;
                ItemBean bean = null;
                RequestDispetcher  rd = null

                try
                {
                        int     itemID = request.getParameter("itemID");
                        bean = new ItemBean();

                        String query = "INSERT INTO Item VALUES(?, ?)";
                        Connection con = this.getConnection();
                        String itemID = request.getParameter("ItemID");
                        PreparedStatement stmt = con.prepareStatement(query);

                        stmt.setInt(1, Integer.valueOf(itemID));
                        stmt.setString(2, request.getParameter("ItemName"));
```

```
                    stmt.executeUpdate();

                    query = "SELECT * FROM Item Where ItemID = " + itemID;
                    stmt = con.prepareStatement(query);
                    stmt.executeQuery();

                    if (rs.next()) {
                        bean.setItemID(Integer.parseInt(rs.getString("ItemID")));
                        bean.setItemName(rs.getString("ItemName"));
                    }

                    stmt.close();
                    con.close();
                }
                catch (Exception e) {
                    out = response.getWriter();
                    out.println("<b>An Error Occurred While Inserting Data </b>");
                }
            }
%>

<%
    this.accessDatabase();
    ItemBean itemBean  = (ItemBean) request.getAttribute("itemBean");
%>

<b> Item Inserted Successfully</b>

<table>
<tr>
    <td>Item ID</td>
    <td>
        <jsp:getProperty name = "itemBean" property = "itemID" >
    </td>
</tr>
<tr>
    <td>Item Name</td>
    <td>
        <jsp:getProperty name = "itemBean"  property = "itemName">
    </td>
</tr>
</table>
</body>
</html>
```