Derby.jar in IntelliJ

Add jar file in “lib” folder

“derby-10.13.1. jar”

Configure of Database

Goto “File” Menu

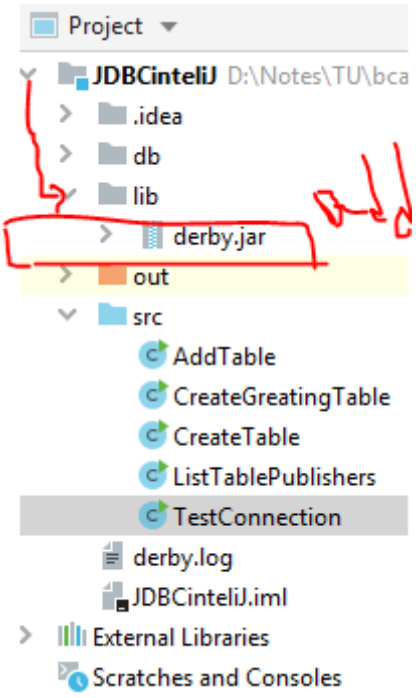
Goto “ Project Structure”

Goto “ Project Settings” -> Select “ Modules”

Then Select “Dependencies” Tab

Then Click on (+) on right side to add “ Jar or directories” Option & “derby.jar”

Apply -| Click on Ok.



1. CreateTable.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class CreateTable {

    public static void main(String[] args) throws Exception {

        Connection con = DriverManager.getConnection("jdbc:derby:D:\\Notes\\TU\\bca 6 th
sem\\db\\testdb;create=true");
        //System.out.println("Connected to Derby Database!");
        Statement st = con.createStatement();
        st.executeUpdate("create table publishers( publisher_id char(6),name char(30), url
```

```
char(80) )");
        System.out.println("table created");
        st.close();
        con.close();


    }

}
```

2. AddTable.java

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

public class AddTable {

    public static void main(String[] args) throws Exception {

        Connection con = DriverManager.getConnection("jdbc:derby:D:\\Notes\\TU\\bca 6 th
sem\\db\\testdb;create=true");
        //System.out.println("Connected to Derby Database!");

        PreparedStatement ps = con.prepareStatement("insert into Publishers(publisher_id, name,
url) values (?,?,?)");
        ps.setString(1, "0201");
        ps.setString(2, "Addison-wesleey");
        ps.setString(3, "www.aw-bc.com");

        ps.executeUpdate();
        ps.close();


        System.out.println("table inserted");

        con.close();


    }

}
```

3. ListTablePublishers.java // no need db connection

```
import java.sql.Connection;
import java.sql.DriverManager;

import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetFactory;
import javax.sql.rowset.RowSetProvider;

public class ListTablePublishers {

    public static void main(String[] args)  throws Exception{
        // TODO Auto-generated method stub
```

```
        RowSetFactory factory = RowSetProvider.newFactory();
        CachedRowSet crs = factory.createCachedRowSet();

        String url = "jdbc:derby:D:\\Notes\\TU\\bca 6 th sem\\db\\testdb";
        crs.setUrl(url);
        crs.setCommand("select * from Publishers");
        crs.execute();
        while(crs.next())
            System.out.println(crs.getString("url"));

        crs.close();


    }

}
```

4. TestConnection.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class TestConnection {

      public static void main(String[] args) throws SQLException {

            Connection conn = DriverManager.getConnection("jdbc:derby:D:\\Notes\\TU\\bca 6 th
sem\\db\\testdb;create=true");
            System.out.println("Connected to Derby Database!");

            Statement stmt = conn.createStatement();

            //Creating a table in Derby database
            String query = "CREATE TABLE Books( "
               + "Title CHAR(60), "
               + "ISBN CHAR(13), "
               + "Publisher_Id CHAR(6), "
               + "Price DECIMAL(10,2))";
            stmt.execute(query);
            stmt.close();

            System.out.println("Table created");
            conn.close();
      }

}
```

5. CreateGreatingTable.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateGreatingTable {

    public static void main(String[] args) throws SQLException {
        // TODO Auto-generated method stub
        Connection conn = DriverManager.getConnection("jdbc:derby:D:\\Notes\\TU\\bca 6 th
sem\\db\\testdb;create=true");
        System.out.println("Connected to Derby Database!");


        Statement stmt = conn.createStatement();
        //Creating a table in Derby database


      String query = "CREATE TABLE Greetings(Message CHAR(20))";
         stmt.execute(query);

          query = "INSERT INTO Greetings VALUES('Hello, World')";
            stmt.execute(query);
         stmt.close();
         System.out.println("Table created");


        ResultSet rs = stmt.executeQuery("Select * from Greetings");
        System.out.println("Contents of the table Greetings table:");
        while(rs.next()) {
            System.out.print(rs.getString("Message"));
            System.out.println();
        }

        conn.close();
    }

}
```

```
import java.nio.file.*;
import java.sql.*;
import java.io.*;
import java.util.*;

/**
 * This program tests that the database and the JDBC driver are correctly configured.
 *
 */
public class TestConnection
{
    public static void main(String args[]) throws IOException
    {
        try
        {
            runTest();
        }
        catch (SQLException ex)
        {
            for (Throwable t : ex)
                t.printStackTrace();
        }
    }

    /**
     * Runs a test by creating a table, adding a value, showing the table contents, and
     removing
     * the table.
     */

    public static void runTest() throws SQLException, IOException
    {
        try (Connection conn = getConnection())
        {
            Statement stat = conn.createStatement();

            stat.executeUpdate("CREATE TABLE Greetings (Message CHAR(20))");
            stat.executeUpdate("INSERT INTO Greetings VALUES ('Hello, World!')");


stat.executeUpdate("CREATE TABLE Publishers (Publisher_Id CHAR(6), Name CHAR(30), URL CHAR(80))");
stat.executeUpdate("INSERT INTO Publishers VALUES ('0201', 'Addison-Wesley', 'www.aw-bc.com')");
stat.executeUpdate("INSERT INTO Publishers VALUES ('0471', 'John Wiley & Sons', 'www.wiley.com')");

String query;

  query = "CREATE TABLE Books( "
         + "Title CHAR(60), "
         + "ISBN CHAR(13), "
         + "Publisher_Id CHAR(6), "
         + "Price DECIMAL(10,2))";

            stat.executeUpdate(query);







  query = "CREATE TABLE BooksAuthors( "
       + "ISBN CHAR(13), "
       + "Author_Id CHAR(6), "
         + "Seq_No CHAR(6))";

            stat.executeUpdate(query);

  query = "CREATE TABLE Authors( "
       + "Author_Id CHAR(6), "
       + "Name CHAR(30), "
```

```
          + "Fname CHAR(30))";

             stat.executeUpdate(query);
             System.out.println("table created");


             stat.executeUpdate("INSERT INTO Authors VALUES ('ALEX', 'Alexander', 'Christopher')");
             stat.executeUpdate("INSERT INTO Authors VALUES ('BROO', 'Brooks', 'Frederick P.')");
             stat.executeUpdate("INSERT INTO Authors VALUES ('ADDI', 'Wesley', 'Addison')");


             stat.executeUpdate("INSERT INTO Books VALUES ('A Guide to the SQL Standard', '0-201-96426-0',
'0201', '47.95')");
             stat.executeUpdate("INSERT INTO Books VALUES ('A Pattern Language: Towns, Buildings', '0-19-
501919-0', '019', '65.00')");


             stat.executeUpdate("INSERT INTO BooksAuthors VALUES ('0-201-96426-0', 'DATE', '1')");
             stat.executeUpdate("INSERT INTO BooksAuthors VALUES ('0-201-96426-0', 'DARW', '2')");
             stat.executeUpdate("INSERT INTO BooksAuthors VALUES ('0-19-501919-0', 'ALEX', '1')");

             System.out.println("inserted");

 try (ResultSet result = stat.executeQuery("SELECT * FROM Publishers"))
 {
 if (result.next())
 System.out.println(result.getString("Publisher_Id"));
 }


 stat.executeUpdate("DROP TABLE Greetings");
          }
    }

    /**
     * Gets a connection from the properties specified in the file database.properties.
     * @return the database connection
     */
    public static Connection getConnection() throws SQLException, IOException
    {
        Properties props = new Properties();
        try (InputStream in = Files.newInputStream(Paths.get("lib/database.properties")))
        {
            props.load(in);
        }
        String drivers = props.getProperty("jdbc.drivers");
        if (drivers != null) System.setProperty("jdbc.drivers", drivers);
        String url = props.getProperty("jdbc.url");
        String username = props.getProperty("jdbc.username");
        String password = props.getProperty("jdbc.password");

        return DriverManager.getConnection(url, username, password);
    }
}
```

Create folder “lib” in Project

```
             // lib/“database.properties” (File)

             jdbc.drivers=org.postgresql.Driver
             jdbc.url=jdbc:postgresql:test
             jdbc.username=postgres
             jdbc.password=postgres
```

Add jar file in “lib” folder

“postgresql-42.2.8.jar”

Configure of Database

Goto “File” Menu

Goto “ Project Structure”

Goto “ Project Settings” -> Select “ Modules”

Then Select “Dependencies” Tab

Then Click on (+) on right side to add “ Jar or directories” Option & “postgresql-42.2.8.jar”

Apply -| Click on Ok.

```
 1          import java.sql.Connection;
 2          import java.sql.DriverManager;
 3          import java.sql.ResultSet;
 4          import java.sql.SQLException;
 5          import java.sql.Statement;
 6
 7  public class TestConnection {
 8
 9      public static void main(String[] args) throws SQLException {
10          // TODO Auto-generated method stub
11          Connection conn = DriverManager.getConnection("jdbc:derby:D:\\Notes\\TU\\bca 6 th
12  sem\\Java Advance Programming\\JDBCinteliJ\\db\\testdb;create=true");
13          System.out.println("Connected to Derby Database!");
14
15
16          Statement stmt = conn.createStatement();
17          //Creating a table in Derby database
18
19
20        String query;
21           query = "CREATE TABLE Greetings (Message CHAR(20))";
22           stmt.execute(query);
23
24            query = "INSERT INTO Greetings VALUES('Hello, World')";
25            stmt.execute(query);
26           stmt.close();
27           System.out.println("Table created");
28
29
30          ResultSet rs = stmt.executeQuery("Select * from Greetings");
31          System.out.println("Contents of the table Greetings table:");
32          while(rs.next()) {
33              System.out.print(rs.getString("Message"));
34              System.out.println();
35          }
36
37
38
39          conn.close();
40      }
41
42  }
43
44
```

45 http://db.apache.org/derby/derby_downloads.html: **db-derby-10.13.1.1-bin**

46 **http://db.apache.org/derby/releases/release-10.13.1.1.html :** db-derby-10.13.1.1-bin.zip

47 **Download:** db-derby-10.13.1.1-bin.zip

48 Extra folder -> goto lib folder -> check "derby.jar" & " derbyclient.jar" files.

49 **Steps: Configure derby db**

50 **1. Open Eclipse IDE "Java EE IDE"**

2. Create new Java Project eg: "JDBC Project"

3. Right Click "Build Path"

4. Select "Configure Build Path"

5. goto "Libraries" tab in "Build path"

6. Add External JAR

7. Choose to folder "derby.jar"

8. Apply

Program 1: Program to check if a file or directory physically exist or not.

```
// In this program, we accepts a file or directory name from
// command line arguments. Then the program will check if
// that file or directory physically exist or not and
// it displays the property of that file or directory.


import java.io.File;

// Displaying file property
class fileProperty
{
    public static void main(String[] args) {
        //accept file name or directory name through command line args
        String fname =args[0];

        //pass the filename or directory name to File object
        File f = new File(fname);

        //apply File class methods on File object
        System.out.println("File name :"+f.getName());
        System.out.println("Path: "+f.getPath());
        System.out.println("Absolute path:" +f.getAbsolutePath());
        System.out.println("Parent:"+f.getParent());
        System.out.println("Exists :"+f.exists());
        if(f.exists())
        {
            System.out.println("Is writeable:"+f.canWrite());
            System.out.println("Is readable"+f.canRead());
            System.out.println("Is a directory:"+f.isDirectory());
            System.out.println("File Size in bytes "+f.length());
        }
    }
}
```

Output:

```
File name :file.txt
Path: file.txt
Absolute path:C:\Users\akki\IdeaProjects\codewriting\src\file.txt
Parent:null
Exists :true
Is writeable:true
Is readabletrue
Is a directory:false
File Size in bytes 20
```

ImageWrite Image in MySQL database

```
import java.awt.image.BufferedImage;
import java.awt.image.RenderedImage;
```

```
import java.io.*;
import java.nio.file.*;
import java.sql.*;
import java.util.*;
import javax.imageio.ImageIO;


/**
 *
 * @author sital
 */
public class ImageWrite {
    public static void main(String args[]) throws IOException
    {
        try
        {
            runTest();
        }
        catch (SQLException ex)
        {
            for (Throwable t : ex)
                t.printStackTrace();
        }
    }

    public static void runTest() throws SQLException, IOException
    {
        try (Connection conn = getConnection())
        {
            System.out.println("connect succ");


            PreparedStatement ps = conn.prepareStatement("insert into players values(?,?)");

            String fname ="src\\javaapplication1\\rocket.jpg";
            File picfile = new File(fname);
            FileInputStream is = new FileInputStream(picfile);

            System.out.println("File name :"+picfile.getName());
            System.out.println("Absolute path:" +picfile.getAbsolutePath());

            ps.setString(1,picfile.getName());
            ps.setBinaryStream(2,is,(int) picfile.length());

            ps.executeUpdate();

        }
    }

    public static Connection getConnection() throws SQLException, IOException
    {
        Properties props = new Properties();
        try (InputStream in = Files.newInputStream(Paths.get("src/lib/database.properties")))
        {
            props.load(in);
        }
        String drivers = props.getProperty("jdbc.drivers");

        if (drivers != null) System.setProperty("jdbc.drivers", drivers);
        String url = props.getProperty("jdbc.url");
        String username = props.getProperty("jdbc.username");
```

```
        String password = props.getProperty("jdbc.password");

        return DriverManager.getConnection(url, username, password);
    }

}
```

src/lib/database.properties

jdbc.drivers=com.mysql.cj.jdbc.Driver

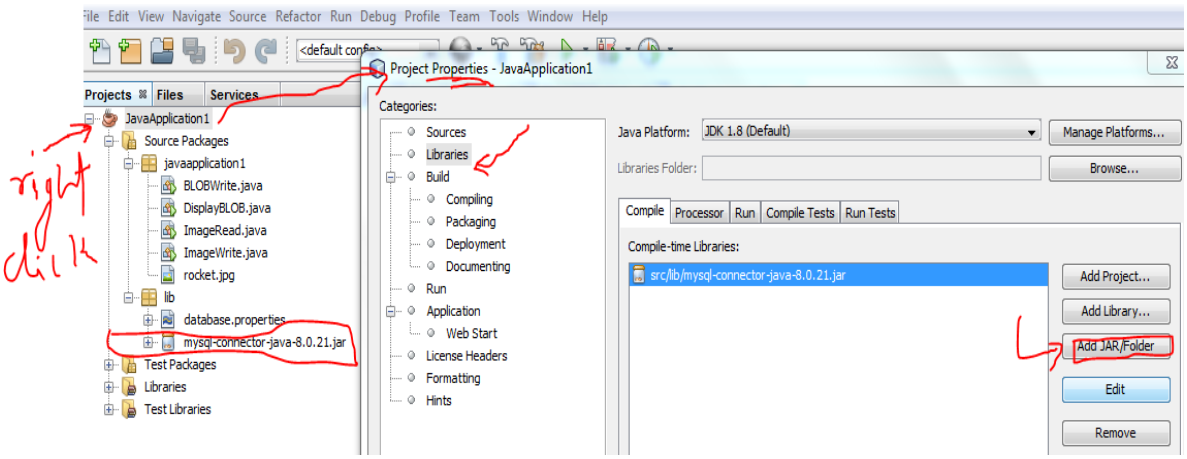
jdbc.url=jdbc:mysql://localhost:3306/test **("test" is an database name)**

jdbc.username=test

jdbc.password=test

add jar file in "lib" folder

mysql-connector-java-8.0.21.jar



// Display Image

```
import java.awt.Image;
import java.awt.Toolkit;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;
import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 *
 * @author sital
 */
public class ImageRead  extends JFrame {

    static JLabel lblimage = new JLabel();
    public static void main(String[] args) throws SQLException, IOException {

        ImageRead window = new ImageRead();

        try (Connection conn = getConnection())
        {

            PreparedStatement stat = conn.prepareStatement("SELECT cover from bookcovers where
isbn=?");
            String isbn = "rocket.jpg";
            stat.setString(1,isbn);
            ResultSet result = stat.executeQuery();

            if(result.next()){
                Blob coverBlob = result.getBlob(1);
                Image coverImage = ImageIO.read(coverBlob.getBinaryStream());

                Icon img = new ImageIcon(coverImage);
                lblimage.setIcon(img);
                window.add(lblimage);
            }
        }

        window.setVisible(true);
        window.pack();
    }


    public static Connection getConnection() throws SQLException, IOException
    {
        Properties props = new Properties();
        try (InputStream in = Files.newInputStream(Paths.get("src/lib/database.properties")))
        {
            props.load(in);
        }
        String drivers = props.getProperty("jdbc.drivers");

        if (drivers != null) System.setProperty("jdbc.drivers", drivers);
```

```
        String url = props.getProperty("jdbc.url");
        String username = props.getProperty("jdbc.username");
        String password = props.getProperty("jdbc.password");

        return DriverManager.getConnection(url, username, password);
    }
}
```

https://www.youtube.com/watch?v=CKQoGo2vUG0

------- ---------- ---------- Servlet Configure ---------- ----------

Copy the file "**servlet-api.jar**" from location **YOUR_INSTILLATION_PATH\tomcat\lib\servlet-api.jar** and Paste the file into your Java Directory **YOUR_INSTILLATION_PATH\Java\jdk1.8.0_121\jre\lib\ext**

this will work (tested).

---------- ---------- Example to config Servlet ----------

Copy : C:\xampp\tomcat\lib\servlet-api.jar

Paste: C:\Program Files\Java\jdk1.8.0_261\jre\lib\ext\servlet-api.jar

------- ---------- ---------- ---------- ---------- ----------

----- ---------- ---------- Example 1 ---------- ---------- ------

C:\xampp\tomcat\webapps\first

//My.java

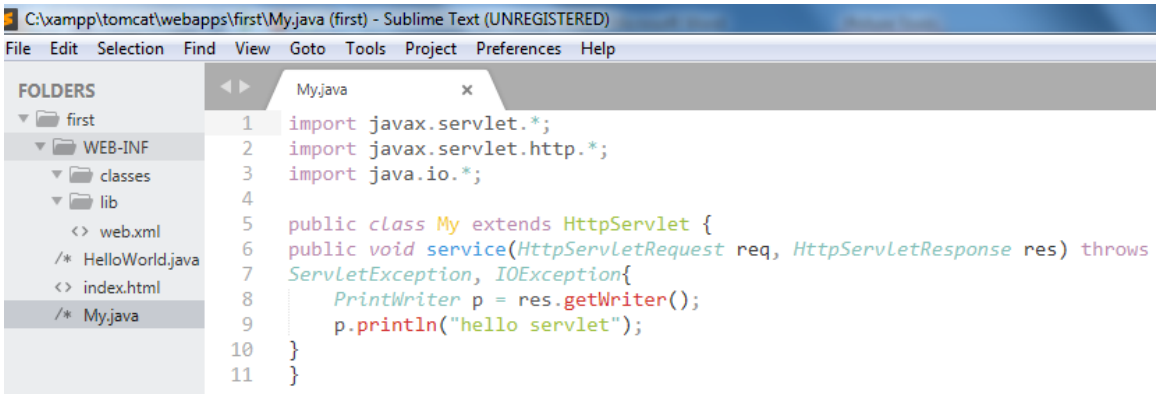
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class My extends HttpServlet {
public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException{
        PrintWriter p = res.getWriter();
        p.println("hello servlet");
}
}
```

---------- ---------- Compile ----------

Class file will store in folder WEB-INF\classes as file name My.class

Javac –d WEB-INF\classes My.java

---------- -



C:\xampp\tomcat\webapps\first\WEB-INF

// WEB-INF/web.xml

<web-app>

<servlet>

<servlet-name>My</servlet-name>

<servlet-class>My</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>My</servlet-name>

<url-pattern>/My</url-pattern>

</servlet-mapping>

</web-app>

----- ---------- ---------- Example 2 ---------- ---------- ------

Servlets are Java classes which service HTTP requests and implement the **javax.servlet.Servlet** interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Sample Code

Following is the sample source code structure of a servlet example to show Hello World −

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

   private String message;

   public void init() throws ServletException {
      // Do required initialization
      message = "Hello World";
   }

   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
```

```
      // Set response content type
      response.setContentType("text/html");

      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
   }

   public void destroy() {
      // do nothing.
   }
}
```

Compiling a Servlet

Let us create a file with name HelloWorld.java with the code shown above. Place this file at C:\ServletDevel (in Windows) or at /usr/ServletDevel (in Unix). This path location must be added to CLASSPATH before proceeding further.

Assuming your environment is setup properly, go in **ServletDevel** directory and compile HelloWorld.java as follows −

```
$ javac HelloWorld.java
```

If everything goes fine, above compilation would produce **HelloWorld.class** file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

Servlet Deployment

By default, a servlet application is located at the path <Tomcat-installationdirectory>/webapps/ROOT and the class file would reside in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class.

For now, let us copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
   <servlet-name>HelloWorld</servlet-name>
   <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>HelloWorld</servlet-name>
   <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

Above entries to be created inside <web-app>...</web-app> tags available in web.xml file. There could be various entries in this table already available, but never mind.

You are almost done, now let us start tomcat server using <Tomcat-installationdirectory>\bin\startup.bat (on Windows) or <Tomcat-installationdirectory>/bin/startup.sh (on Linux/Solaris etc.) and finally type **http://localhost:8080/HelloWorld** in the browser's address box. If everything goes fine, you would get the following result

Reference Web:

https://www.informit.com/articles/article.aspx?p=26920&seqNum=4

https://www.javatpoint.com/steps-to-create-a-servlet-using-tomcat-server

https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html

https://www.javatpoint.com/example-of-login-form-in-servlet