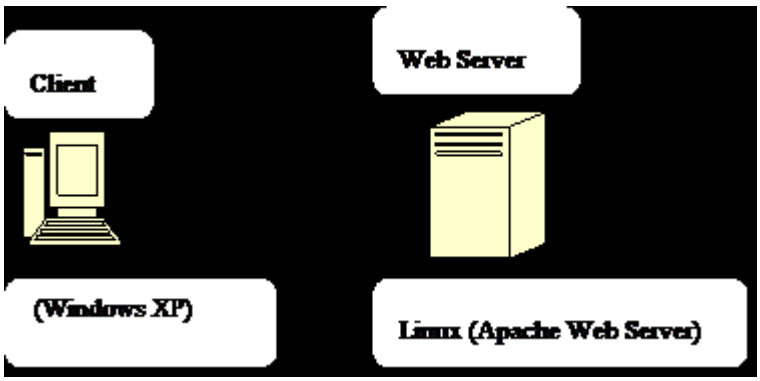
JSP tutorial

What is JSP?

JSP-JavaServer Pages is a web technology that bases on Java programming language. JSP is used to build dynamic web sites. JSP pages are HTML pages embedded Java code to generate dynamic content. To learn JSP, you should already learn at least Java programming and HTML.

When you type a jsp page (the file with extension: .jsp) in a web browser of client machine computer (e.g. Firefox, IE, or Google Chrome), the web browser will send a request to the web server (e.g. Apache Tomcat, or JRUN) and the web server will use servlet engine to generate a html file and send the html file output to the browser.

JSP is easy to learn and allows web developers to build dynamic web sites and web applications quickly. Web sites built with JSP can run on many platforms such as Apache Tomcat, JRUN, Caucho Resin, or Jetty. And they also can run on many operating systems (Windows, Linux…).



What will you need?

Before you can run your Jsp web pages, you need to install Java Development Toolkit (jdk1.6.0_25) and Tomcat 6.0 or newer. You may use search engines to get these packages and instructions to install them.

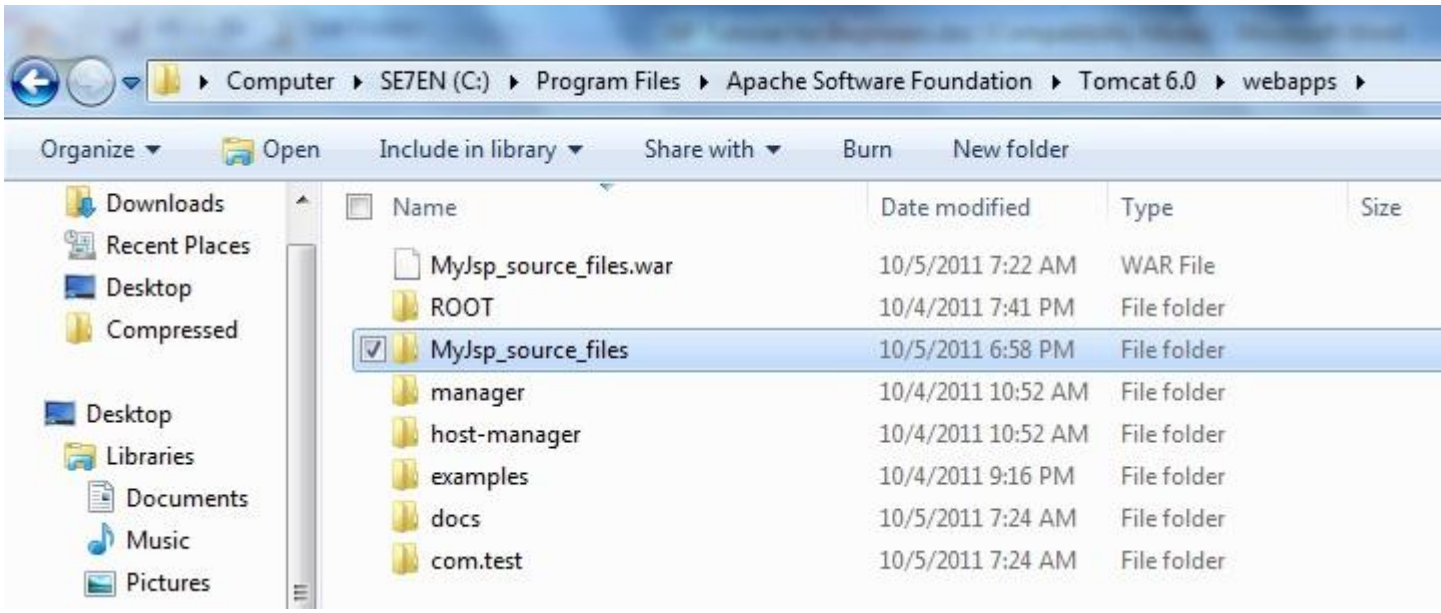
Your first Jsp program

Now, it is the time to start writing a Jsp page. Open a text editor (e.g. Notepad) and type the following code.

```
<html>

<head>

  <title>My first Jsp page</title>

</head>

<body>

<!--Your Jsp code   -->
```

```
<%@ page language="java" %>

<% out.println("<H1>Jsp is powerful.</H1>"); %>

</body>

</html>
```

Save the file as myfirst.jsp in a folder (e.g. MyJsp_source_files) that locates in the default web server directory (C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps).



Open your web browser and type:

http://localhost:8080/MyJsp_source_files/myfirst.jsp

Using Declaration tag (<%! %>)

In the declaration tag, you can declare variables and methods.

Example:

```
<%!

String yourname="Dara Yuk";

String getName(String name){return name;}

%>
```

Using Scriptlet tag(<% %>)

With Scriptlet tag, you can write your Java code to access variables declared in declaration tag.

Example:

Welcome.jsp

```
<html>

<body>

<%!

String yourname="Dara Yuk";

String getName(String name){

 return name;

}

 %>


<% out.println("Welcome "+getName(yourname)+"!"); %>


</body>

</html>
```

Using Expression tag (<%= %>)

In the expression tag, you can embed Java expression.

Example:

```
<%=new java.util.Date() %>
```

Using Directive tag--<%@ include file %>

You can use <%@include file %> directive tag to include html file or jsp in to you jsp file.

Example: in this example, we have three files: Logo.htm, DisplayCurrentDate.jsp, and Welcome.jsp. We will include Logo.htm and DisplayCurrentDate.jsp in to Welcome.jsp. Logo.htm

will display an image referring our web page logo displayed on the top Welcome.jsp page and DisplayCurrentDate.jsp page will display below other text.

Logo.htm

```
<html>

<body>

<img src="images/image442.gif"  alt="Logo" height="72" width="184" />

</body>


</html>
```

DisplayCurrentDate.jsp

```
<html>

<body>

<%=new java.util.Date() %>

</body>

</html>
```

Welcome.jsp

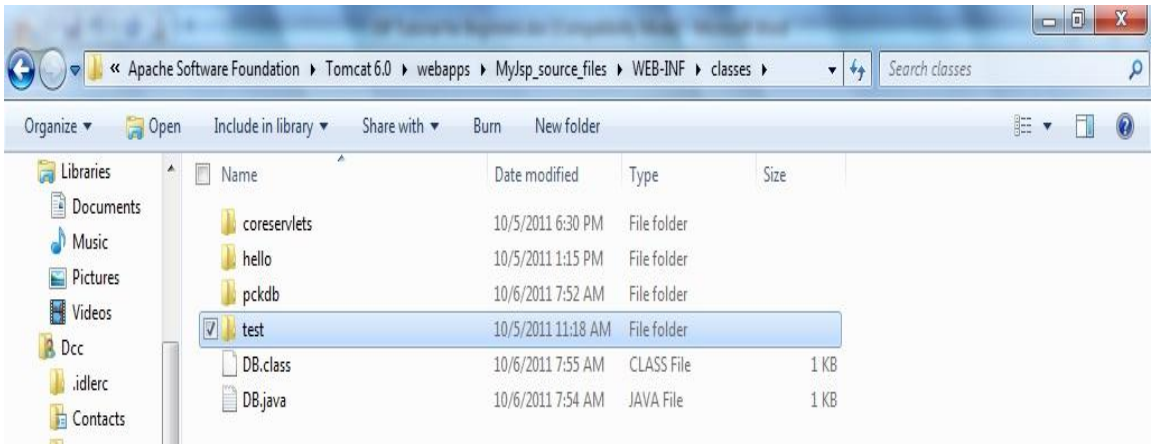
```
<html>

<head>

  <title>Welcome Page</title>

</head>

<body style="Background-color:silver">


<!--Logo.htm is included here   -->

<%@ include file="Logo.htm" %>
```

```
<% out.println("<H1>Welcome to our learning center!</H1>"); %>

<%@ include file="DisplayCurrentDate.jsp" %>



</body>

</html>
```

Using Directive tag--<%@page import %>

With <%@page import %> directive tag, you can import Java classes in to your jsp page.

Example: in this example, we will import a class called Employee contained in Employee.java of the test folder. It is important to note that the test folder must be put in WEB-INF/classes directory of your default Tomcat server directory and Employee.java must be compiled to generate Employee.class file. If you are not sure to compile a java file, you may read our Java tutorial: Java Tutorial for Beginners



Using Directive tag--<%@taglib %>

<%@taglib uri="tag descriptor file" prefix="tag prefix" %>

You can create a new collection of html tags and call them to work in a jsp page without writing Java code in scriptlets. With a large amount of Java code, scriptplets are messy. Therefore, you should you use custom tags instead of scriptlets.

To create a custom tag, you will create two files: Java file (.java) to handle the tag actions, and tag descriptor file (.tld) to describe the new tag in xml form.

Example: in this example, we will create a new tag “example” to display a “Tag example…” message.

TagExample.java (WEB-INF/classes/tags/TagExamle.java)

```
package tags;

import javax.servlet.jsp.*;

import javax.servlet.jsp.tagext.*;

import java.io.*;

/* A simple tag to print "Tag example..." into the output.

*/

public class TagExample extends TagSupport {

public int doStartTag() {

try {

JspWriter out = pageContext.getOut();

out.print("Tag example... ");

} catch(IOException ioe) {

System.out.println("Error in ExampleTag: " + ioe);

}

return(SKIP_BODY);

}

}
```

Note: TagExample.java file must be compiled to TagExample.class. On important thing is you must specify the serverlet.jar package in classpath environment when you compile the TagExample.java file, otherwise, you will get error messages (e.g. javax.servelet.jsp does not exist). If you don't have servlet.jar package, click here to download. And don't forget to put it in the tags folder containing TagExample.java file. If you don't put it in the tags folder, you need to specify its correct path when you compile the TagExample.java file. The following command might help you compile the TagExample.java:

javac –cp servlet.jar TagExample.java

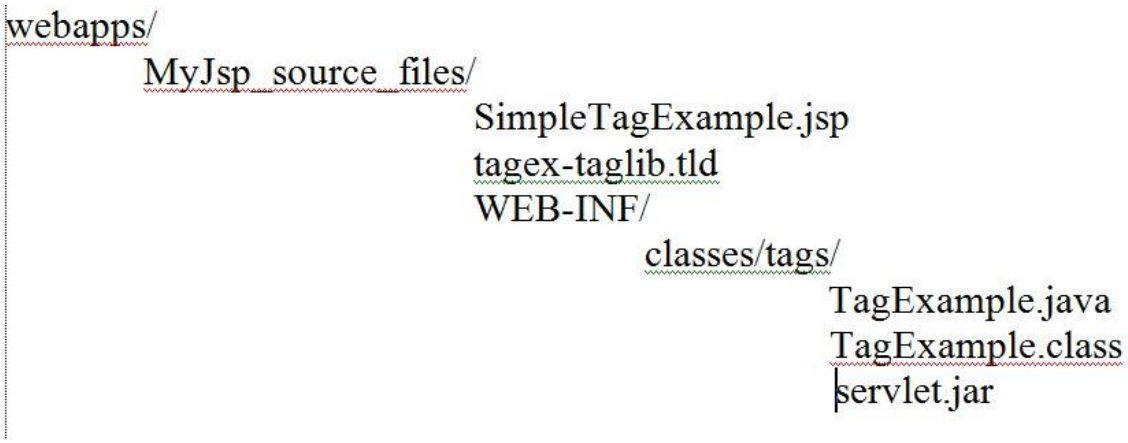
tagex-taglib.tld(located in the folder with jsp file)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<!-- a tag library descriptor -->
<taglib>

<tlibversion>1.0</tlibversion>
<jspversion>1.1</jspversion>
<shortname>tagex</shortname>
<urn></urn>
<info>
A custom tag library
</info>
<tag>
<!--Tag name defined here... -->
<name>example</name>
<!--Specify the Java class that handles the tag action... -->
<tagclass>tags.TagExample</tagclass>
<info>Simple tag example: print one line to the output</info>
<bodycontent>EMPTY</bodycontent>
</tag>
<!-- Other tags defined later... -->
</taglib>
```

SimpleTagExample.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>

<HEAD>

<%@ taglib uri="tagex-taglib.tld" prefix="tagex" %>

<TITLE>Custom Tag Example</TITLE>

</HEAD>

<BODY>

<H1><tagex:example /></H1>

<tagex:example />

</BODY>

</HTML>
```

See the below folders hierarchy that might help you organize your files in this example:



Using JavaBeans with Jsp

JavaBeans is a special Java class that has specific methods to be called to work in Jsp pages. It is commonly used to handle data input by the use in html form. The data input in the form will be store in the beans. Therefore, the data can be used later in other jsp pages. The most important thing about JavaBeans is the code written in the beans can be reusable.

Example: In this example, we will ask the user to input his/her name and e-mail on DataCollection.jsp page. Once the use clicks on submit button, the data will be stored in a bean called UserInfo class located in WEB-INF/classes/userpack folder and then these information will be processed in a Thank.jsp.

UserInfo.java (WEB-INF/classes/userpack/UserInfo.java)

```
package userpack;
public class UserInfo{
private String youremail;
private String yourname;
public UserInfo() {
yourname = null;
youremail=null;
}
public void setYourname( String name ) {
yourname = name;
}
public String getYourname() {
return yourname;
}

public void setYouremail( String email ) {
youremail = email;
}
public String getYouremail() {
return youremail;
}

}
```

Remind: Don’t forget to compile UserInfo.java to UserInfo.class file.

DataCollection.jsp

```
<%@ page import="userpack.UserInfo" %>
<jsp:useBean id="beanName" scope="page" class=" userpack.UserInfo " />
<jsp:setProperty name="beanName" property="yourname" />
<jsp:useBean id="beanEmail" scope="page" class=" userpack.UserInfo " />
<jsp:setProperty name="beanEmail" property="youremail" />

<html>
<head><title>User Data Collection</title></head>
<body bgcolor="#FF00C0" >

<table border="0" width="600">
<tr>
<td width="140">   </td>
<td width="540">
<h1>Please enter your name and e-mail</h1>
</td>
</tr>
<tr>
<td width="140"   </td>
<td width="540">
<form method="get">
<table>
```

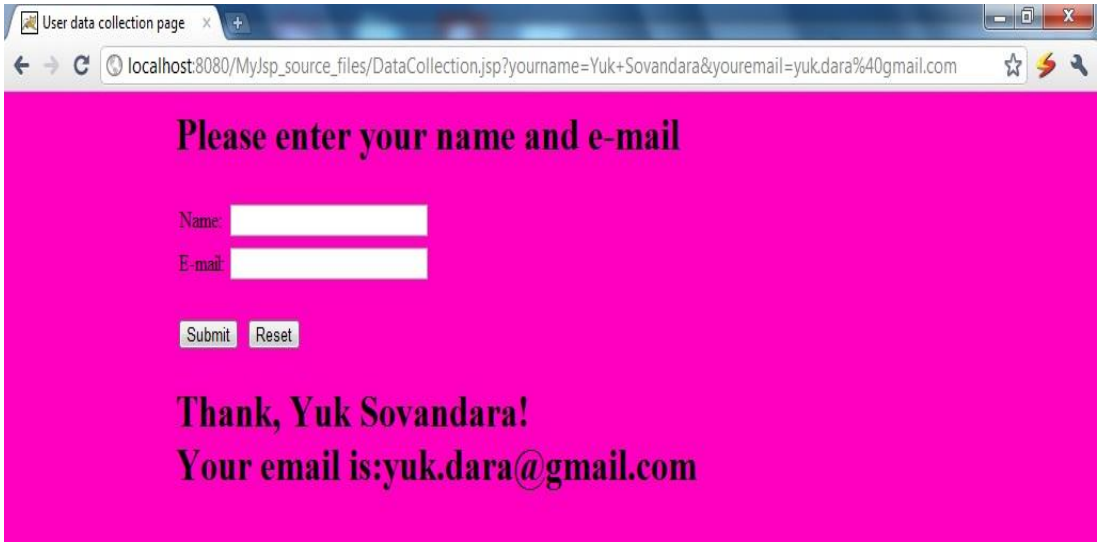
```
<tr><td>Name:</td>
<td><input type="text" name="yourname" size="20"></td>
</tr>
<tr> <td>E-mail:</td>
<td><input type="text" name="youremail" size="20"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</td>
</tr>
</form>
</table>
<%
if ( request.getParameter("yourname") != null ) {
%>
<%@ include file="Thank.jsp" %>
<%
}
%>
</body>
</html>
```

Thank.jsp

```
<html>
```

```
<body>

<table border="0" width="600">

<tr>

<td width="140">

 

</td>

<td width="540">

<h1>Thank, <jsp:getProperty name="beanName" property="yourname" />! <br />

Your email is:<jsp:getProperty name="beanEmail" property="youremail" />

</h1>

</td>

</tr>

</table>

</body>

</html>
```

Note: request.getParameter() is used to get information submitted by post or get method. Post method will hide information in address bar and it does not limit the length of information that is attached to the address. Get method will display information attached with the address in address bar. It limits to the short length of information (100 characters).



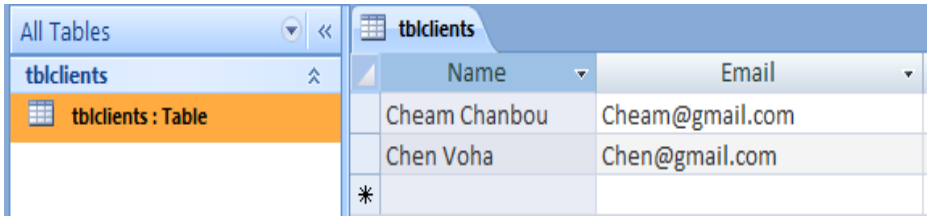
Jsp and Ms. Access Data source

This section might be useful for those who want to build a dynamic web site with data stored in a Ms. Access data source.

Example: In this example, we modify the above example to store clients' information in Ms. Access database called dbclients. Once the uses click submit button, his/her information will be saved in the tblclients table of the database. All clients' information can be accesses through ClientList.jsp file.

Step 1:

Create a database "dbclients" and a table "tblclients". This table has only two fields: Name and Email.



Then create a data source "dbclients" from the database by double-clicking on Data Sources (ODBC) in Administrative Tools. Click Add and select System DSN, then choose Microsoft Access Driver (*.mdb,*.accdb). Click Finish, then type dbstudent in Data Source Name box and click Select…button to browse for the database "dbclients" location.

Step 2:

Now create a Java file called Mydb.java to handle the connectivity, retrieve data, and insert data to the database. Mydb.java will be placed in WEB-INF/classes/pckdb folder.

```
package pckdb;

import java.sql.*;


public class Mydb{

        private  Connection con;

        private Statement statement;

        private  ResultSet rs;

  public Mydb(){
```

```
        try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         con=DriverManager.getConnection("jdbc:odbc:dbclients","","");
         statement=con.createStatement();
         }catch(ClassNotFoundException e){System.out.println("Unable to connect");
         }catch(Exception se){System.out.println("sql error!");}
  }

 public String displayWithbrowser(){
        String dbdata="<table>";
         try{
         rs=statement.executeQuery("Select * From tblclients");
        while(rs.next()){
             dbdata+="<tr>"+"<td style='width:300px'>" + rs.getString(1) + "</td><td
style='width:300px'>" + rs.getString(2) +  "</td></tr>";

        }

  }catch(SQLException se){System.out.println("sql error!");}
  return dbdata+"</table>";
  }

public int insertdata(String name, String email){
         try{
        statement.executeUpdate("INSERT INTO tblclients(Name,Email)
Values('"+name+"','"+email+"')");
```

```
    }catch(SQLException se){System.out.println("sql error!");}
    return 1;
 }

public void closecon(){
try{
con.close();
   }catch(SQLException se){System.out.println("sql error!");}

  }
}
```

Modify the Thank.jsp file to include the following code:

```
<%@ page import="pckdb.Mydb" %>
<%
Mydb dbclient=new Mydb();
if(dbclient.insertdata(request.getParameter("yourname"),request.getParameter("youremail"))==1){
    out.println("Your information has been recorded.");}
    else out.println("Unable to insert data!");
    dbclient.closecon();

 %>
```

Thank.jsp

```
<html>
```

```
<body>
<table border="0" width="700">
<tr>
<td width="150">
 
</td>
<td width="550">
<h1>Thank, <jsp:getProperty name="beanName" property="yourname" />! <br />
Your email is:<jsp:getProperty name="beanEmail" property="youremail" />
</h1>
</td>
</tr>
</table>
<%@ page import="pckdb.Mydb" %>
<% Mydb dbclient=new Mydb();

if(dbclient.insertdata(request.getParameter("yourname"),request.getParameter("youremail"))==1){
    out.println("Your information has been recorded.");}
    else out.println("Unable to insert data!");
    dbclient.closecon();

%>

</body>
</html>
```

Next create a ClientList.jsp file to get data from the data source and a Header.html file to display data:

ClientList.jsp

```
<%@ page import="pckdb.Mydb" %>
<html>
<body style="Background-color:#CCC0FF">

<% out.println("<h2>Clients List</h2>"); %>
<%@ include file="Header.html" %>
<h3>---------------------------------------------------------------------------</h3>
<% Mydb dbclient=new Mydb();
   out.println(dbclient.displayWithbrowser());
   dbclient.closecon();
%>
<h3>---------------------------------------------------------------------------</h3>
</body>
</html>
```

Header.html

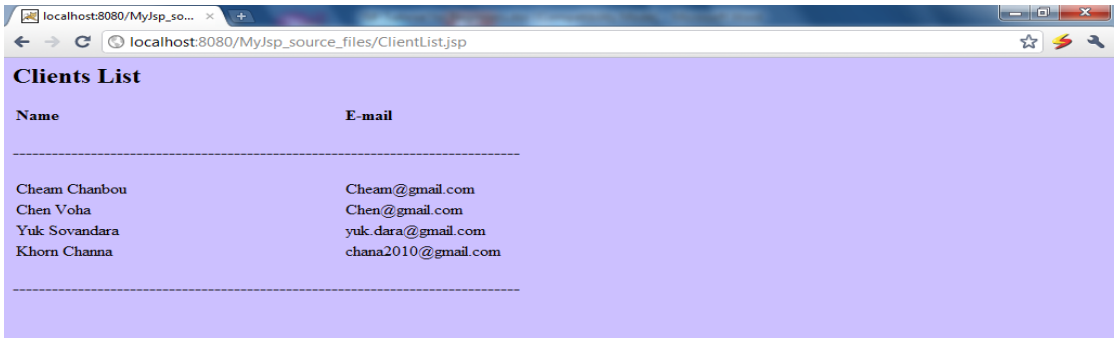
```
<html>

<head>
<style type="text/css">
.style1 {
        text-align: left;
}
</style>
</head>
```

```
<body>
<table>
<tr>
<th  style="width:300px" class="style1">Name </th>
<th style="width: 300px" class="style1">E-mail</th>
</tr>
</table>
</body>
</html>
```

Finally, open your browser to see the results:

By pushing the Submit button, you'll get:

To view all clients' information, open the ClientList.jsp page:



Jsp and Mysql database

Mysql is one of the most popular database management software for web and desktop applications. In this section, we will modify Mydb.java file used in the previous example to connect and get data from dbclients database in Mysql database server.

Step 1: Login to Mysql server in your local machine and create dbclients database and tblclients as you did in the previous example. If you do not have Mysql server installed in your local machine, please install it before you go to step 2.

-Login to Mysql local server by root user:

>myslq –h localhost –u root

if your Mysql local server requires password to log in, use can use the command:

>mysql –h localhost –u root –p (then enter the password)

-Create database dbclients:

mysql> CREATE DATABASE dbclients;

-Create dbclients table:

mysql> USE dbclients;

mysql> CREATE TABLE tblclients(Name varchar(20),Email varchar(40));

-Insert sample data:

mysql>INSERT INTO tblclients(Name,Email)

Values('Yuk Dara','yuk.dara@gmail.com');

```
mysql>INSERT INTO tblclients(Name,Email)

        Values('Khorn Channa','Khorn2010@gmail.com');
```

-View the data

```
mysql>SELECT * FROM tblclients;
```

Step 2:

Now, you have dbclients database with sample data in Mysql server. Next, you only need to modify Mydb.java file by replacing two lines:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

con=DriverManager.getConnection("jdbc:odbc:dbclients","","");
```

With:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();

con = DriverManager.getConnection("jdbc:mysql:///dbclients","root",null);
```

Step 3:

Compile Mydb.java again and restart Tomcat server service. Then open your browser and test.