Author：相忠良
Email: ugoood@163.com

起始于：June 7, 2018
最后更新日期：June 8, 2018

声明：本笔记依据传智播客方立勋老师 **Java Web** 的授课视频内容记录而成，中间加入了自己的理解。本笔记目的是强化自己学习所用。若有疏漏或不当之处，请在评论区指出。谢谢。
涉及的图片，文档写完后，一次性更新。

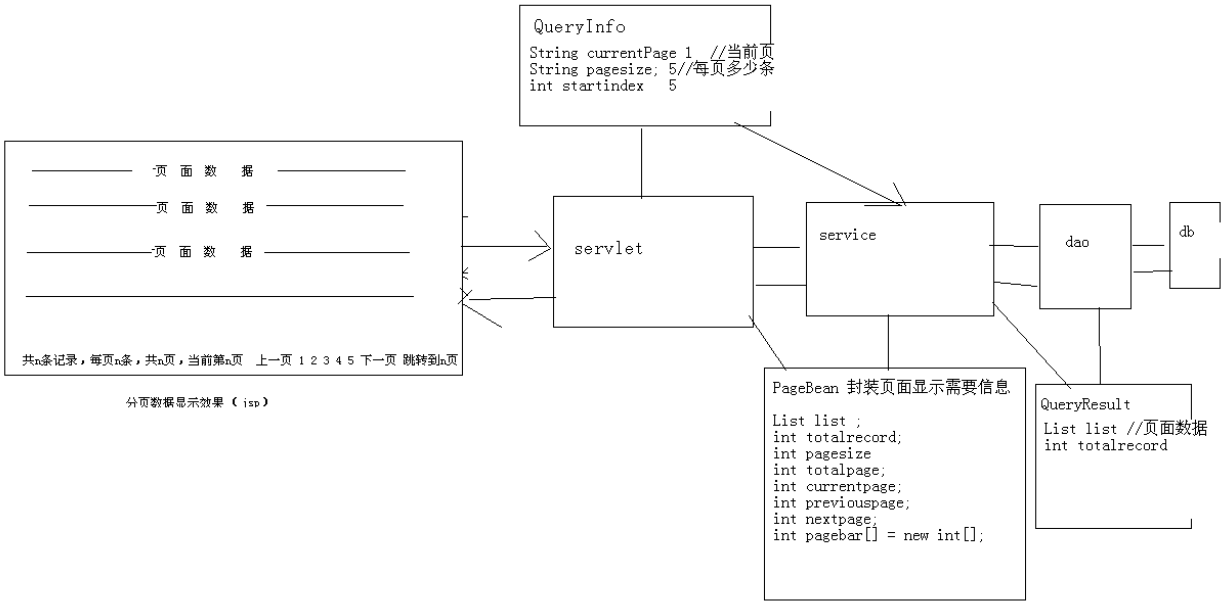
day15 分页及 JDBC 大数据的处理

本节案例是承接 day14 的客户关系管理系统，继续改造。

1. 分页实现(重点)

所需小知识，如下：
`select id,name from customer limit 0,5;` 显示第1-5条数据，第1页；
`select id,name from customer limit 5,5;` 显示第6-10条数据，第2页；
`select id,name from customer limit 10,5;` 显示第11-15条数据，第3页。

分页结构图如下(重要的就是这个图，后面的工作均围绕这个图展开)：



上图涉及了3个封装数据的 javabean，先实现了他们！如下：

`cn.wk.domain.PageBean`：

```java
package cn.wk.domain;

import java.util.List;

public class PageBean {
    private List list;
    private int totalrecord;
    private int pagesize;
    private int totalpage;
    private int currentpage;
    private int previouspage;
    private int nextpage;
    private int[] pagebar;

    public int getPagesize() {
        return pagesize;
    }

    public void setPagesize(int pagesize) {
        this.pagesize = pagesize;
    }

    public List getList() {
        return list;
    }

    public void setList(List list) {
        this.list = list;
    }

    public int getTotalrecord() {
        return totalrecord;
    }

    public void setTotalrecord(int totalrecord) {
        this.totalrecord = totalrecord;
    }

    public int getTotalpage() {
        if (this.totalrecord % this.pagesize == 0)
            this.totalpage = this.totalrecord / this.pagesize;
        else
            this.totalpage = this.totalrecord / this.pagesize + 1;
        return totalpage;
    }

    public int getCurrentpage() {
        return currentpage;
    }
```

```
    public void setCurrentpage(int currentpage) {
        this.currentpage = currentpage;
    }

    public int getPreviouspage() {
        if (this.currentpage - 1 < 1)
            this.previouspage = 1;
        else
            this.previouspage = this.currentpage - 1;
        return previouspage;
    }

    public int getNextpage() {
        if (this.currentpage + 1 >= this.totalpage)
            this.nextpage = this.totalpage;
        else
            this.nextpage = this.currentpage + 1;
        return nextpage;
    }

    public int[] getPagebar() {
        int startpage;
        int endpage;
        int pagebar[] = null;
        if (this.totalpage <= 10) {
            pagebar = new int[this.totalpage];
            startpage = 1;
            endpage = this.totalpage;
        } else {
            pagebar = new int[10];
            startpage = this.currentpage - 4;
            endpage = this.currentpage + 5;

            if (startpage < 1) {
                startpage = 1;
                endpage = 10;
            }

            if (endpage > this.totalpage) {
                endpage = this.totalpage;
                startpage = this.totalpage - 9;
            }
        }

        int index = 0;
        for (int i = startpage; i <= endpage; i++) {
            pagebar[index++] = i;
        }
        this.pagebar = pagebar;
        return this.pagebar;
    }
```

```
}
```

`cn.wk.domain.QueryInfo`：

```
package cn.wk.domain;

public class QueryInfo {
    private int currentpage = 1; // 用户要看的页
    private int pagesize = 5;    // 用户想看的页面大小
    private int startindex;      // 记住用户看的页的数据在数据库的起始位置

    public int getCurrentpage() {return currentpage;}
    public void setCurrentpage(int currentpage) {
        this.currentpage = currentpage;
    }
    public int getPagesize() {return pagesize;}
    public void setPagesize(int pagesize) {this.pagesize = pagesize;}
    public int getStartindex() {
        this.startindex = (this.currentpage - 1) * this.pagesize;
        return startindex;
    }
}
```

`cn.wk.domain.QueryResult`：

```
package cn.wk.domain;

import java.util.List;

public class QueryResult {
    private List list;       // 记住用户看的页的数据
    private int totalrecord; // 记住总记录数

    public List getList() {return list;}
    public void setList(List list) {this.list = list;}
    public int getTotalrecord() {return totalrecord;}
    public void setTotalrecord(int totalrecord) {
        this.totalrecord = totalrecord;
    }
}
```

`cn.wk.dao.impl.CustomerDaoImpl`添加1个方法(同时将该方法声明加入`cn.wk.dao.CustomerDao`接口中)，如下：

```
/* 为分页功能 所添加的代码 */
// 获取页面数据和总记录数
public QueryResult pageQuery(int startindex, int pagesize) {

  Connection conn = null;
  PreparedStatement st = null;
  ResultSet rs = null;

  QueryResult qr = new QueryResult();

  try {
    conn = JdbcUtils.getConnection();
    String sql = "select * from customer limit ?,?";
    st = conn.prepareStatement(sql);
    st.setInt(1, startindex);
    st.setInt(2, pagesize);
    rs = st.executeQuery();
    List list = new ArrayList();
    while (rs.next()) {
      Customer c = new Customer();
      c.setBirthday(rs.getDate("birthday"));
      c.setCellphone(rs.getString("cellphone"));
      c.setDescription(rs.getString("description"));
      c.setEmail(rs.getString("email"));
      c.setGender(rs.getString("gender"));
      c.setId(rs.getString("id"));
      c.setName(rs.getString("name"));
      c.setPreference(rs.getString("preference"));
      c.setType(rs.getString("type"));
      list.add(c);
    }
    qr.setList(list);

    // 总记录数
    sql = "select count(*) from customer";
    st = conn.prepareStatement(sql);
    rs = st.executeQuery();
    if (rs.next()) {
      qr.setTotalrecord(rs.getInt(1));
    }
    return qr;
  } catch (Exception e) {
    throw new DaoException(e);
  } finally {
    JdbcUtils.release(conn, st, rs);
  }
}
```

修改`cn.wk.service.impl.BusinessServiceImpl`，向其添加下面的方法(同时将该方

法声明加入`cn.wk.service.BusinessService`接口中)，如下：

```
public PageBean pageQuery(QueryInfo queryInfo) {
  // 调用 dao 获取页面数据
  QueryResult qr = dao.pageQuery(queryInfo.getStartindex(),
      queryInfo.getPagesize());

  // 根据 dao 查询结果，生成页面显示所需的 pagebean
  PageBean bean = new PageBean();
  bean.setCurrentpage(queryInfo.getCurrentpage());
  bean.setList(qr.getList());
  bean.setPagesize(queryInfo.getPagesize());
  bean.setTotalrecord(qr.getTotalrecord());
  return bean;
}
```

修改`cn.wk.web.controller.ListCustomerServlet`如下：

```java
package cn.wk.web.controller;

import java.io.IOException;
import java.util.List;

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

import cn.wk.domain.PageBean;
import cn.wk.domain.QueryInfo;
import cn.wk.service.BusinessService;
import cn.wk.service.impl.BusinessServiceImpl;
import cn.wk.utils.WebUtils;

// 用分页技术，得到所有客户显示
public class ListCustomerServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        try {
            // 把 request 信息封装成 查询信息 bean 中
            QueryInfo info = WebUtils.request2Bean(req, QueryInfo.class);
            BusinessService service = new BusinessServiceImpl();
            PageBean pagebean = service.pageQuery(info);
            req.setAttribute("pagebean", pagebean);
            req.getRequestDispatcher("/WEB-INF/jsp/listcustomer.jsp").for
ward(
                    req, resp);
        } catch (Exception e) {
            e.printStackTrace();
            req.setAttribute("message", "查看客户失败！！");
            req.getRequestDispatcher("/message.jsp").forward(req, resp);
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

最后，修改`/WEB-INF/jsp/listcustomer.jsp`(改成分页显示，而不是全部显示)如下：

`<%@ taglib uri="/wk" prefix="wk"%>`在 1.1 节有定义。

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="/wk" prefix="wk"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<title>列出所有客户</title>

<style type="text/css">
    .even{background-color:#FFFFCC}
    .odd{background-color:#CCFFFF}
    tr:hover{background-color:#FFCCCC}
</style>
</head>

<body style="text-align: center;">
    <table frame="border" width="85%">
        <tr>
            <td>客户姓名</td>
            <td>性别</td>
            <td>生日</td>
            <td>手机</td>
            <td>邮箱</td>
            <td>爱好</td>
            <td>类型</td>
            <td>备注</td>
            <td>操作</td>
        </tr>

        <c:forEach var="c" items="#{requestScope.pagebean.list}" varStatu
s="status">
            <tr class="${status.count%2==0?'even':'odd'}">
                <td>${c.name }</td>
                <td>${c.gender }</td>
                <td>${c.birthday }</td>
                <td>${c.cellphone }</td>
                <td>${c.email }</td>
                <td>${wk:sub(c.preference) }</td>
                <td>${c.type }</td>
                <td>${wk:sub(c.description) }</td>
                <td>
                    <a href="${pageContext.request.contextPath}/servlet/E
ditCustomerServlet?id=${c.id}">修改</a>
                    <a href="javascript:void(0)" onclick="del(${c.id})">删
除</a>
                    <!-- <a href="javascript:void(0)" onclick="del('${c.i
d}')">删除</a>  加单引号才能删，怕数据被别人删，所以注释掉 -->
```

```
                </td>
            </tr>
        </c:forEach>
    </table>
    <br />

    <script type="text/javascript">
        function del(id){
            if(window.confirm("您确定删除？？")){
                window.location.href='${pageContext.request.contextPath}/
servlet/DelCustomerServlet?id=' + id;
            }
        }

        function gotopage(currentpage) {
            if (currentpage < 1 || currentpage != parseInt(currentpage)
                    || currentpage > ${pagebean.totalpage}) {
                alert("请输入有效值！！");
                document.getElementById("pagenum").value = '';
            } else {
                var pagesize = document.getElementById("pagesize").value;
                window.location.href = '${pageContext.request.contextPat
h}/servlet/ListCustomerServlet?currentpage='
                        + currentpage + '&pagesize=' + pagesize;
            }
        }
    </script>

    共[${pagebean.totalrecord}]条记录，
    每页
    <input type="text" id="pagesize" value="${pagebean.pagesize}"
        onchange="gotopage(${pagebean.currentpage})" style="width:30px"
        maxlength="2">条，
     共[${pagebean.totalpage}]页，
    当前第[${pagebean.currentpage}]页    

    <a href="javascript:void(0)"
        onclick="gotopage(${pagebean.previouspage})">上一页</a>

    <c:forEach var="pagenum" items="${pagebean.pagebar}">
        <c:if test="${pagenum==pagebean.currentpage }">
            <font color="red">${pagenum}</font>
        </c:if>

        <c:if test="${pagenum!=pagebean.currentpage }">
            <a href="javascript:void(0)" onclick="gotopage(${pagenum})">
${pagenum}</a>
        </c:if>
    </c:forEach>

    <a href="javascript:void(0)" onclick="gotopage(${pagebean.nextpag
```

```
e})">下一页</a>

    <input type="text" id="pagenum" style="width:35px" >
    <input type="button" value=" GO " onclick="gotopage(document.getEleme
ntById('pagenum').value)">

</body>
</html>
```

1.1 自建 EL 表达式去处理简介过长问题

`cn.wk.utils.MyEL`，注意 EL 需建立静态方法，代码如下：

```
package cn.wk.utils;

public class MyEL {
    public static String sub(String str) {
        if (str.length() > 10)
            return str.substring(0, 10) + "......";
        return str;
    }
}
```

再写一个tld文件，`\WEB-INF\wk.tld`：

```
<?xml version="1.0" encoding="UTF-8" ?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
  version="2.0">

  <description>JSTL 1.1 functions library</description>
  <display-name>JSTL functions</display-name>
  <tlib-version>1.1</tlib-version>
  <short-name>fn</short-name>
  <uri>/wk</uri>

   <function>
    <name>sub</name>
    <function-class>cn.wk.utils.MyEL</function-class>
    <function-signature>java.lang.String sub(java.lang.String)</function-signature>
  </function>

</taglib>
```

2. 完成客户关系管理案例

修改客户信息的`cn.wk.web.controller.EditCustomerServlet`：

```java
package cn.wk.web.controller;

import java.io.IOException;

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

import cn.wk.domain.Customer;
import cn.wk.service.BusinessService;
import cn.wk.service.impl.BusinessServiceImpl;
import cn.wk.utils.Globals;
import cn.wk.utils.WebUtils;

public class EditCustomerServlet extends HttpServlet {

    // 根据id获取要修改的客户信息
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        String id = req.getParameter("id");
        BusinessService service = new BusinessServiceImpl();
        Customer c = service.findCustomer(id);

        req.setAttribute("genders", Globals.genders);
        req.setAttribute("preferences", Globals.preferences);
        req.setAttribute("types", Globals.types);

        req.setAttribute("c", c);
        req.getRequestDispatcher("/WEB-INF/jsp/editcustomer.jsp").forward
(req,
                resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        req.setCharacterEncoding("UTF-8");
        // 把填写的表单修改信息封装到 customer 对象中
        try {
            Customer c = WebUtils.request2Bean(req, Customer.class); // 里
面有id
            BusinessService service = new BusinessServiceImpl();
            service.updateCustomer(c);
            req.setAttribute("message", "更新成功！！");
        } catch (Exception e) {
```

```
            e.printStackTrace();
            req.setAttribute("message", "更新失败！！");
        }
        req.getRequestDispatcher("/message.jsp").forward(req, resp);
    }
}
```

`/WEB-INF/jsp/editcustomer.jsp`如下(涉及数据回显):

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>修改用户的视图</title>
    <script type="text/javascript" src="${pageContext.request.contextPath }/js/ShowCalendar.js"></script>
    <script type="text/javascript">
        function makepre(){
            var pres = document.getElementsByName("pre");
            var preference = "";
            for(var i = 0; i < pres.length; i++){
                var input = pres[i];
                if(input.checked==true){
                    preference = preference + input.value + ",";
                }
            }
            // 组装字符串    跳舞,打麻将,看凤姐
            preference = preference.substr(0, preference.length - 1);

            var form = document.getElementById("form");
            var input = document.createElement("input");
            input.type = "hidden";
            input.name = "preference";
            input.value = preference;

            form.appendChild(input);
            return true;
        }
    </script>
  </head>

<body style="text-align:center;">
    <br />
    <form id="form"
        action="${pageContext.request.contextPath}/servlet/EditCustomerServlet"
        method="post" onsubmit="return makepre()">
        <!-- js代码，当按submit按钮时，就调用makepre()方法 -->

        <table border="1" width="30%">
            <input type="hidden" name="id" value="${c.id}">
            <tr>
                <td>客户姓名</td>
                <td><input type="text" name="name" value="${c.name}"></td>
            </tr>
```

```
            <tr>
                <td>性别</td>
                <td><c:forEach var="gender" items="${genders}">
                        <input type="radio" name="gender" value="${gende
r}" ${c.gender==gender?'checked':'' }> ${gender}
                </c:forEach></td>
            </tr>

            <tr>
                <td>生日</td>
                <td><input type="text" name="birthday"
                    onClick="showCalendar(this.id)" id="birthday" value
="${c.birthday}"></td>
            </tr>

            <tr>
                <td>手机</td>
                <td><input type="text" name="cellphone" value="${c.cellph
one}"></td>
            </tr>

            <tr>
                <td>邮箱</td>
                <td><input type="text" name="email" value="${c.email}"></
td>
            </tr>

            <tr>
                <td>爱好</td>
                <td><c:forEach var="p" items="${preferences}">
                        <input type="checkbox" name="pre" value
="${p}" ${fn:contains(c.preference,p)?'checked':'' }>${p}
                </c:forEach></td>
            </tr>

            <tr>
                <td>客户类型</td>
                <td><c:forEach var="t" items="${types}">
                        <input type="radio" name="type" value="${t}" ${c.
type==t?'checked':''}>${t}
                </c:forEach></td>
            </tr>

            <tr>
                <td>客户备注</td>
                <td>
                <textarea rows="5" cols="100" name="description">${c.desc
ription}</textarea>
                </td>
            </tr>
```

```
            <tr>
                <td><input type="reset" value="重置"></td>
                <td><input type="submit" value="修改客户"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

删除客户数据的`cn.wk.web.controller.DelCustomerServlet`：

```
package cn.wk.web.controller;

import java.io.IOException;

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

import cn.wk.service.BusinessService;
import cn.wk.service.impl.BusinessServiceImpl;

// 删除记录
public class DelCustomerServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        try {
            String id = req.getParameter("id");
            BusinessService service = new BusinessServiceImpl();
            service.deleteCustomer(id);
            req.setAttribute("message", "删除成功！！");
        } catch (Exception e) {
            e.printStackTrace();
            req.setAttribute("message", "删除失败！！");
        }
        req.getRequestDispatcher("/message.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

3. jdbc 大数据的处理

准备：新建 java 工程，名字为 day15。

这里的大数据 LOB(Large Objects) 仅仅指：

- clob 用于存大文本，Text；
- blob 用于存二进制数据，如图像，声音，二进制文件等。

对MySQL而言只有blob，而没有clob，mysql存储大文本采用的是Text，Text和blob分别又分为：

- TINYTEXT、TEXT、MEDIUMTEXT和LONGTEXT
- TINYBLOB、BLOB、MEDIUMBLOB和LONGBLOB

对于MySQL中的Text类型，可调用如下方法去存：

```
// 给个reader流，而不能给String，大数据的处理只能通过流
PreparedStatement.setCharacterStream(index, reader, length);
//注意length长度须设置，并且设置为int型
```

对MySQL中的Text类型，可调用如下方法获取：

```
reader = resultSet. getCharacterStream(i);
reader = resultSet.getClob(i).getCharacterStream();
string s = resultSet.getString(i);
```

例子 Demo1：
前提是准备好 资源文件db.properties，1.txt，mysql-connector-java-5.0.8-bin.jar，创建好day15数据库和 testclob 表(在下面代码里有)，然后在做这个实验。

```
public class Demo1 {

    /**
     * 读写大文本     *
     * create table testclob ( id varchar(40) primary key, resume text );
     * */

    @Test
    public void insert() throws SQLException, FileNotFoundException {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "insert into testclob(id,resume) values(?,?)";
            st = conn.prepareStatement(sql);
            st.setString(1, "1");

            File file = new File("src/1.txt");
            FileReader reader = new FileReader(file);
            st.setCharacterStream(2, reader, (int) file.length());
            int num = st.executeUpdate();
            if (num > 0)
                System.out.println("插入成功！！");
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

  @Test
    public void read() throws SQLException, IOException {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "select id, resume from testclob where id='1'";
            st = conn.prepareStatement(sql);
            rs = st.executeQuery();

            if (rs.next()) {
                // String resume = rs.getString("resume"); 不能这样做，内存
会崩

                Reader reader = rs.getCharacterStream("resume"); //<---重
点

                FileWriter writer = new FileWriter("e:\\1.txt");
                try {
```

```
                    int len = 0;
                    char buffer[] = new char[1024];
                    while ((len = reader.read(buffer)) > 0) {
                        writer.write(buffer, 0, len);
                    }
                } finally {
                    if (reader != null) {
                        reader.close();
                    }
                    if (writer != null) {
                        writer.close();
                    }
                }
            }
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }
}
```

db.properties如下：

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/day15
username=root
password=root
```

对于MySQL中的BLOB类型，可调用如下方法设置：

```
PreparedStatement. setBinaryStream(i, inputStream, length);
```

对MySQL中的BLOB类型，可调用如下方法获取：

```
InputStream in  = resultSet.getBinaryStream(i);
InputStream in  = resultSet.getBlob(i).getBinaryStream();
```

实验所用的表：

```
create table testclob ( id varchar(40) primary key, image blob );
```

细节，略。

4. jdbc 实现数据库批处理

小知识： `truncate table testbatch;` 清除 testbatch 中的所有数据。
下面例子展现了 **oracle** 的强大。插入 **10000006** 条记录，**mysql** 用大约**3**小时，而 **oracle** 只用 **380**秒，即 **6** 分多钟！

例子(展示了2种批处理的方式)：
第二种方式适合做批量插入和更新，而第一种方式可发不同种的sql语句。实际开发中，第二种用的多！

```java
package cn.itcast.demo;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.junit.Test;

import cn.itcast.utils.JdbcUtils;

public class Demo3 {

    /*
     create table testbatch
     (
        id varchar(40) primary key,
        name varchar(40)
     );
    */

    //实现批处理第一种方式
    @Test
    public void test1() throws SQLException{
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;

        try{
            conn = JdbcUtils.getConnection();
            String sql1 = "insert into testbatch(id,name) values('1','aa
a')";
            String sql2 = "update testbatch set name='bbb' where id='1'";

            st = conn.createStatement();  //list
            st.addBatch(sql1);
            st.addBatch(sql2);

            //[3,4]
            st.executeBatch();
            st.clearBatch();

        }finally{
            JdbcUtils.release(conn, st, rs);
        }
    }


    //实现批处理的第二种方式
```

```
    @Test
    public void test2() throws SQLException{

        long starttime = System.currentTimeMillis();
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try{
            conn = JdbcUtils.getConnection();
            String sql = "insert into testbatch(id,name) values
(?,?)";   //作批量插入   批量更新
            st = conn.prepareStatement(sql);

            for(int i=1;i<=10000006;i++){
                st.setString(1, i+"");
                st.setString(2, "aa" + i);
                st.addBatch();
                if(i%1000==0){
                    st.executeBatch(); // 每1000个sql做成批，向数据库发一次
                    st.clearBatch(); // 清除 st 维护的 list 中的数据
                }
            }
            st.executeBatch(); // 剩余的部分做成批，最后再向数据库发一次
        }finally{
            JdbcUtils.release(conn, st, rs);
        }

        long endtime = System.currentTimeMillis();
        System.out.println("总花了: " + (endtime-starttime)/1000 + "秒");
    }
}
```

5. jdbc 获取数据库自动生成的主键和调用存储过程

5.1 数据库自动生成的主键

```
public class Demo4 {
    /**
     获取自动生成的主键
     use day15;
     create table test(
        id int primary key auto_increment,
        name varchar(40)
     );
     */
    public static void main(String[] args) throws SQLException {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "insert into test(name) values('aaa')";

            // 设置st是否能获取自动生成的主键      但下面的设置不起任何作用，不知为啥
            st = conn.prepareStatement(sql, Statement.NO_GENERATED_KEYS); //<--
            st.executeUpdate();
            rs = st.getGeneratedKeys(); // <-- 重点
            if(rs.next()){
                System.out.println(rs.getInt(1));
            }
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }
}
```

5.2 jdbc 调用存储过程(procedure) - 金融证券领域用的特多

金融证券领域的开发无法使用 hibernate，因为数据库表结构是保密的，只有存储过程暴露在外。此领域只能通过 jdbc 调用存储过程去获取数据。

存储过程就是数据库那面的方法或者函数。处理数据用。现在，我们有两种方式从数据库中获取我们想要的数据：

1. 在数据库中编写存储过程，由该 procedure 把处理后的数据给我们；
2. 直接把数据库的数据拉到我们这来，我们通过 java 编写函数来处理数据，最终获取我们想要的数据。

银行的业务，如利息是在数据库中用存储过程来实现的，而不是在数据库外用其他语言如 **java**，由其他 **coder** 实现！

试验准备，编写一个存储过程，如下：

```
use day15;
delimiter $$

CREATE PROCEDURE demoSp(IN inputParam VARCHAR(255), INOUT inOutParam varc
har(255))
BEGIN
    SELECT CONCAT('zyxw---', inputParam) into inOutParam;
END $$

delimiter ;
```

得到CallableStatement，并调用存储过程：

```
CallableStatement cStmt = conn.prepareCall("{call demoSp(?, ?)}");
```

设置参数，注册返回值，得到输出：

```
cStmt.setString(1, "abcdefg");
cStmt.registerOutParameter(2, Types.VARCHAR);
cStmt.execute();
System.out.println(cStmt.getString(2));
```

完整例子：

```
public class Demo5 {
    public static void main(String[] args) throws SQLException {
        Connection conn = null;
        CallableStatement cStmt = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            cStmt = conn.prepareCall("{call demoSp(?,?)}");

            cStmt.setString(1, "haha");
            cStmt.registerOutParameter(2, Types.VARCHAR);
            cStmt.execute();
            System.out.println(cStmt.getString(2));
        } finally {
            JdbcUtils.release(conn, cStmt, rs);
        }
    }
}
```

6. ResultSet 对结果集进行滚动

ResultSet 提供了对结果集进行滚动的方法：

- next()：移动到下一行
- Previous()：移动到前一行
- absolute(int row)：移动到指定行
- beforeFirst()：移动resultSet的最前面。
- afterLast() ：移动到resultSet的最后面

可对小数据量内容分页，不可应用于大数据量，因为数据量大时，ResultSet 对象会很大。应该象第 1 节分页实现里讲的那样，你要那些数据，就查询哪些数据进行显示，而不是一次性把所有数据拿回来，封装到 ResultSet 对象中再处理，这种方式若数据量足够大，内存会崩！