**[java数据库编程——读写LOB、可滚动和可更新的结果集、元数据](http://www.cnblogs.com/gaopeng527/p/4530168.html)**

**[java 数据库编程](http://blog.csdn.net/seacean2000/article/details/9937755)**

**1. 读写LOB**

　　除了数字、字符串和日期之外，许多数据库还可以存储大对象，例如图片或其它数据。在SQL中，二进制大对象称为BLOB，字符型大对象称为CLOB。

　　要读取LOB，需要执行SELECT语句，然后在ResultSet上调用getBlob或getClob方法，这样就可以获得Blob或Clob类型的对象。要从Blob中获取二进制数据，可以调用getBytes或getInputStream。例如，如果你有一张保存图书封面图形的表，那么就可以像下面这样获取一张图像：

PreparedStatement stat = conn.prepareStatement("SELECT Cover FROM BookCovers WHERE ISBN=?");

stat.set(1, isbn);

ResultSet result = stat.executeQuery();

if(result.next()){

Blob coverBlob = result.getBlob(1);

Image coverImage = ImageIO.read(coverBlob.getBinaryStream());

}

　　类似地，如果获取了Clob对象，那么就可以通过调用getSubString或getCharacterStream方法来获取其中的字符数据。

　　要将LOB置于数据库中，需要在Connection对象上调用createBlob或createClob，然后获取一个用于该LOB的输出流或写出器，写出数据，并将该对象存储到数据库中。例如，下面展示了如何存储一张图像：

Blob coverBlob = connection.createBlob();

int offset = 0;

OutputStream out = coverBlob.setBinaryStream(offset); ImageIO.write(coverImage, "PNG", out);

PreparedStatement stat = conn.prepareStatement("INSERT INTO Cover VALUES(?, ?)");

stat.set(1, isbn);

stat.set(2,coverBlob);

stat.executeUpdate();

**2. 可滚动和可更新的结果集**

　　要让ResultSet可以滚动个和更新，必须在创建Statement对象的时候使用下面的方式指定对应的参数：

Statement stmt = conn.createStatement(type, concurrency);

　　对于PreparedStatement，使用下面的方式指定参数：

PreparedStatement pstmt = conn.prepareStatement(sql, type, concurrency);

其中，type表示ResuleSet的类型，而concurrency表示是否可以使用ResuleSet来更新数据库。

**type和concurrency的取值以及含义如下：**

|  |  |  |  |
| --- | --- | --- | --- |
| ResultSet类的type值 | | | |
| 值 | | 解释 | |
| ResultSet.TYPE\_FORWARD\_ONLY | | 结果集不能滚动（默认值） | |
| ResultSet.TYPE\_SCROLL\_INSENSITIVE | | 结果集可以滚动，但ResuleSet对数据库中数据变化不敏感 | |
| ResultSet.TYPE\_SCROLL\_SENSIT IVE | | 结果集可以滚动，并且ResuleSet对数据库中发生的改变敏感 | |
| ResultSet类的Concurrency值 | | |
| 值 | 解释 | |
| ResultSet.CONCUR\_READ\_ONLY | 结果集不能用于更新数据库（默认值） | |
| ResultSet.CONCUR\_UPDATABLE | 结果集可以用于更新数据库 | |

JDBC的结果集有很多类型。这些结果集有不同的特性，以满足各种需要。这在高性能的JDBC数据操作中有着重要应用。下面是一个应用实例：

**package** com.zs.jdbc;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement; /\*\* \* JDBC可滚动可更新感知更新结果集测试 \* \* **@author** leizhimin 2009-12-8 20:09:03 \*/

**public** **class** Test {

**public** **static** **void** main(String[] args){

*testScrollResultSet*();

*testUpdateResultSet*();

} /\*\* \* 可更新结果集更新测试 \*/

**public** **static** **void** testUpdateResultSet()

{

Connection conn = JDBCTest.*getconnection*();

String sql = "SELECT \* FROM book";

**try** {

Statement stmt = conn.createStatement(ResultSet.***TYPE\_SCROLL\_SENSITIVE***, ResultSet.***CONCUR\_UPDATABLE***);

ResultSet rs = stmt.executeQuery(sql);

System.***out***.println("---------原结果集--------");

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

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

}

System.***out***.println("---------插入一条记录--------");

rs.first();

//将光标移动到插入行上

rs.moveToInsertRow();

//构建行数据

rs.updateString(2, "xxxx");

rs.updateString(3, "x");

//插入一行

rs.insertRow();

System.***out***.println("-------------更新一条记录-------------");

rs.absolute(3);

//构建行数据

rs.updateString(2, "uuuu");

rs.updateString(3, "u");

rs.updateRow();

System.***out***.println("---------插入更新后的结果集--------");

rs = stmt.executeQuery(sql);

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

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

}

rs.close();

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

} **finally** {

**try** {

conn.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\* \* 可滚动结果集滚动测试 \*/

**public** **static** **void** testScrollResultSet() {

Connection conn = JDBCTest.*getconnection*();

String sql = "SELECT \* FROM book";

**try** {

Statement stmt = conn.createStatement(ResultSet.***TYPE\_SCROLL\_SENSITIVE***, ResultSet.***CONCUR\_READ\_ONLY***);

ResultSet rs = stmt.executeQuery(sql);

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

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

}

System.***out***.println("------前滚操作-----");

//将光标移动到此 ResultSet 对象的上一行

rs.previous();

rs.previous();

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

System.***out***.println("------绝对定位-----");

//将光标移动到此 ResultSet 对象的给定行编号。

rs.absolute(3);

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

System.***out***.println("------移动到第一行-----");

//将光标移动到此 ResultSet 对象的第一行。

**if** (rs.first()) {

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

}

System.***out***.println("------移动到最后一行-----");

//将光标移动到此 ResultSet 对象的第一行。

**if** (rs.last()) {

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

}

System.***out***.println("------移动到第一行之前-----");

//将光标移动到此 ResultSet 对象的开头，正好位于第一行之前

rs.beforeFirst();

rs.next();

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

System.***out***.println("------移动到最后一行之后-----");

//将光标移动到此 ResultSet 对象的末尾，正好位于最后一行之后。

rs.afterLast();

rs.previous();

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

System.***out***.println("------相对当前行做移动-----");

rs.relative(-2);

System.***out***.println("[行号：" + rs.getRow() + "]\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));

rs.close();

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

} **finally** {

**try** {

conn.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

copycode.gif

控制台输出：

copycode.gif

[行号：1] 1 aaa a [行号：2] 2 bbb b [行号：3] 3 ccc c [行号：4] 4 ddd d [行号：5] 5 eee e [行号：6] 6 fff f [行号：7] 7 ggg g [行号：8] 8 hhh h ------前滚操作----- [行号：7] 7 ggg g ------绝对定位----- [行号：3] 3 ccc c ------移动到第一行----- [行号：1] 1 aaa a ------移动到最后一行----- [行号：8] 8 hhh h ------移动到第一行之前----- [行号：1] 1 aaa a ------移动到最后一行之后----- [行号：8] 8 hhh h ------相对当前行做移动----- [行号：6] 6 fff f ---------原结果集-------- [行号：1] 1 aaa a [行号：2] 2 bbb b [行号：3] 3 ccc c [行号：4] 4 ddd d [行号：5] 5 eee e [行号：6] 6 fff f [行号：7] 7 ggg g [行号：8] 8 hhh h ---------插入一条记录-------- -------------更新一条记录------------- ---------插入更新后的结果集-------- [行号：1] 1 aaa a [行号：2] 2 bbb b [行号：3] 3 uuuu u [行号：4] 4 ddd d [行号：5] 5 eee e [行号：6] 6 fff f [行号：7] 7 ggg g [行号：8] 8 hhh h [行号：9] 9 xxxx x Process finished with exit code 0

copycode.gif

可保存性：设置提交时候是否关闭结果集。

ResultSet.HOLD\_CURSORS\_OVER\_COMMIT ：在提交后结果集还可用

ResultSet.CLOSE\_CURSORS\_AT\_COMMIT：在提交时候关闭结果集

由于这些特性比较高级，不同数据库驱动对此实现也不一样。因此在使用JDBC高级特性的时候最好做个测试，以保证程序的可靠性。

当type设置为：ResultSet.TYPE\_SCROLL\_INSENSITIVE 或者 ResultSet.TYPE\_SCROLL\_INSENSITIVE 时，游标可以移动，但是移动的位置是[1，count]，记住并不是从0开始，否则会报错。

既然可以移动，那么把移动的几个方法解释一下：

rs = statement.executeQuery();  游标指向第一行前面的位置，这个位置是不能获取数据，否则报错：结果集没有当前行

rs.next();  // 游标下移一个位置，如果所在位置有结果集那么返回true，否则返回false

rs.previous(); // 游标上移一个位置，如果所在位置有结果集那么返回true，否则返回false

rs.first(); //  游标指向第一行的位置

rs.last(); //  游标指向最后一行的位置

rs.beforeFirst(); // 游标指向第一行前面的位置 ， 这个位置不能获取数据

rs.afterLast(); //  游标指向最后一行后面的位置，这个位置不能获取数据

rs.absolute(index); // 游标移动至index位置，index是[1,count]的任意数字，但是不能超出，否则报错

rs.relative(index); // 游标从当前位置算移动index个位置，也就是相对移动，index可以是负数，但是计算结果同样在[1,count]内

isAfterLast(); // 判断游标是否在最后一行之后。

isBeforeFirst();// 判断游标是否在第一行之前。

ifFirst() ;  //判断游标是否指向结果集的第一行。

isLast(); // 判断游标是否指向结果集的最后一行。

getRow();// 得到当前游标所指向行的行号，行号从1开始，如果结果集没有行，返回0。

**3. 元数据**

　　元数据在SQL中是用来描述数据库或其组成部分的数据。我们可以获得三类元数据：关于数据库的元数据，关于结果集的元数据，关于预备语句参数的元数据。元数据是描述基本数据信息的数据，操作元数据的语法如下：

DatabaseMetaData meta=conn.getMetaData(); ResultSet mrs=meta.getTables(null,null,null,new String[]{"TABLE"});

元数据结果集中第一列代表了表目录，第二列代表了表结构模式，第三列表名，第四列表类型，第五列关于表的注释。

　　下面我们编写一个简单的数据库工具，通过使用元数据来浏览数据库中的所有表，该程序还展示了如何使用带缓存的行集。

方法一：直接连接数据库，代码如下：

copycode.gif

package view; import java.awt.\*; import java.awt.event.\*; import java.io.\*; import java.nio.file.\*; import java.sql.\*; import java.util.\*; import javax.sql.\*; import javax.sql.rowset.\*; import javax.swing.\*; /\*\* \* This program uses metadata to display arbitrary tables in a database. \* \* @author DELL \*/public class ViewDB { public static void main(String[] args) { EventQueue.invokeLater(new Runnable() { public void run() { JFrame frame = new ViewDBFrame(); frame.setTitle("ViewDB"); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setVisible(true); } }); } }/\*\* \* The frame that holds the data panel and the navigation buttons. \*/class ViewDBFrame extends JFrame { private JButton previousButton; private JButton nextButton; private JButton deleteButton; private JButton saveButton; private DataPanel dataPanel; private Component scrollPane; private JComboBox<String> tableNames; private Properties props; private CachedRowSet crs; public ViewDBFrame() { tableNames = new JComboBox<String>(); tableNames.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showTable((String) tableNames.getSelectedItem()); } }); add(tableNames, BorderLayout.NORTH); try{ Class.forName("com.mysql.jdbc.Driver");// readDatabaseProperties(); try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234")) { DatabaseMetaData meta = conn.getMetaData(); ResultSet mrs = meta.getTables(null, null, null, new String[] { "TABLE" }); while (mrs.next()) tableNames.addItem(mrs.getString(3)); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } catch (ClassNotFoundException e) { // TODO Auto-generated catch block e.printStackTrace(); } JPanel buttonPanel = new JPanel(); add(buttonPanel, BorderLayout.SOUTH); previousButton = new JButton("Previous"); previousButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showPreviousRow(); } }); buttonPanel.add(previousButton); nextButton = new JButton("Next"); nextButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showNextRow(); } }); buttonPanel.add(nextButton); deleteButton = new JButton("Delete"); deleteButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { deleteRow(); } }); buttonPanel.add(deleteButton); saveButton = new JButton("Save"); saveButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { saveChanges(); } }); buttonPanel.add(saveButton); pack(); } /\*\* \* Prepares the text fields for showing a new table, and shows the first row. \* @param tableName the name of the table to display \*/ public void showTable(String tableName) { try { Class.forName("com.mysql.jdbc.Driver"); try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234")) { // get result set Statement stat = conn.createStatement(); ResultSet result = stat.executeQuery("SELECT \* FROM " + tableName); // copy into cached row set RowSetFactory factory = RowSetProvider.newFactory(); crs = factory.createCachedRowSet(); crs.setTableName(tableName); crs.populate(result); } if (scrollPane != null) remove(scrollPane); dataPanel = new DataPanel(crs); scrollPane = new JScrollPane(dataPanel); add(scrollPane, BorderLayout.CENTER); validate(); showNextRow(); } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } catch (ClassNotFoundException e) { // TODO Auto-generated catch block e.printStackTrace(); } } /\*\* \* Moves to the previous table row. \*/ public void showPreviousRow() { try { if (crs == null || crs.isFirst()) return; crs.previous(); dataPanel.showRow(crs); } catch (SQLException e) { for (Throwable t : e) t.printStackTrace(); } } /\*\* \* Moves to the next table row. \*/ public void showNextRow() { try { if (crs == null || crs.isLast()) return; crs.next(); dataPanel.showRow(crs); } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } } /\*\* \* Deletes current table row. \*/ public void deleteRow() { try { Class.forName("com.mysql.jdbc.Driver"); try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234")) { conn.setAutoCommit(false); crs.deleteRow(); crs.acceptChanges(conn); if (crs.isAfterLast()) if (!crs.last()) crs = null; dataPanel.showRow(crs); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } catch (ClassNotFoundException e) { // TODO Auto-generated catch block e.printStackTrace(); } } /\*\* \* Saves all changes. \*/ public void saveChanges() { try { Class.forName("com.mysql.jdbc.Driver"); try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","1234")) { conn.setAutoCommit(false); dataPanel.setRow(crs); crs.acceptChanges(conn); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } catch (ClassNotFoundException e) { // TODO Auto-generated catch blocke.printStackTrace(); } }// private void readDatabaseProperties() throws IOException // { // props = new Properties(); // try (InputStream in = Files.newInputStream(Paths.get("database.properties"))) // { // props.load(in); // } // String drivers = props.getProperty("jdbc.drivers"); // if (drivers != null) System.setProperty("jdbc.drivers", drivers); // } /\*\* \* Gets a connection from the properties specified in the file database.properties. \* @return the database connection \*/// private Connection getConnection() throws SQLException // { // String url = props.getProperty("jdbc.url"); // String username = props.getProperty("jdbc.username"); // String password = props.getProperty("jdbc.password"); //// return DriverManager.getConnection(url, username, password); // }}/\*\* \* This panel displays the contents of a result set. \*/class DataPanel extends JPanel { private java.util.List<JTextField> fields; /\*\* \* Constructs the data panel. \* @param rs the result set whose contents this panel displays \*/ public DataPanel(RowSet rs) throws SQLException { fields = new ArrayList<>(); setLayout(new GridBagLayout()); GridBagConstraints gbc = new GridBagConstraints(); gbc.gridwidth = 1; gbc.gridheight = 1; ResultSetMetaData rsmd = rs.getMetaData(); for (int i = 1; i <= rsmd.getColumnCount(); i++) { gbc.gridy = i - 1; String columnName = rsmd.getColumnLabel(i); gbc.gridx = 0; gbc.anchor = GridBagConstraints.EAST; add(new JLabel(columnName), gbc); int columnWidth = rsmd.getColumnDisplaySize(i); JTextField tb = new JTextField(columnWidth); if (!rsmd.getColumnClassName(i).equals("java.lang.String")) tb.setEditable(false); fields.add(tb); gbc.gridx = 1; gbc.anchor = GridBagConstraints.WEST; add(tb, gbc); } } /\*\* \* Shows a database row by populating all text fields with the column values. \*/ public void showRow(ResultSet rs) throws SQLException { for (int i = 1; i <= fields.size(); i++) { String field = rs == null ? "" : rs.getString(i); JTextField tb = fields.get(i - 1); tb.setText(field); } } /\*\* \* Updates changed data into the current row of the row set. \*/ public void setRow(RowSet rs) throws SQLException { for (int i = 1; i <= fields.size(); i++) { String field = rs.getString(i); JTextField tb = fields.get(i - 1); if (!field.equals(tb.getText())) rs.updateString(i, tb.getText()); } rs.updateRow(); } }

copycode.gif

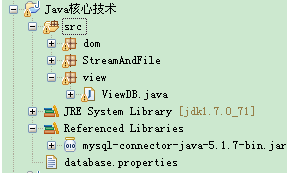
方法二：使用database.properties配置文件，代码如下：

copycode.gif

package view; import java.awt.\*; import java.awt.event.\*; import java.io.\*; import java.nio.file.\*; import java.sql.\*; import java.util.\*; import javax.sql.\*; import javax.sql.rowset.\*; import javax.swing.\*; /\*\* \* This program uses metadata to display arbitrary tables in a database. \* \* @author DELL \*/public class ViewDB { public static void main(String[] args) { EventQueue.invokeLater(new Runnable() { public void run() { JFrame frame = new ViewDBFrame(); frame.setTitle("ViewDB"); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setVisible(true); } }); } }/\*\* \* The frame that holds the data panel and the navigation buttons. \*/class ViewDBFrame extends JFrame { private JButton previousButton; private JButton nextButton; private JButton deleteButton; private JButton saveButton; private DataPanel dataPanel; private Component scrollPane; private JComboBox<String> tableNames; private Properties props; private CachedRowSet crs; public ViewDBFrame() { tableNames = new JComboBox<String>(); tableNames.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showTable((String) tableNames.getSelectedItem()); } }); add(tableNames, BorderLayout.NORTH); try { readDatabaseProperties(); try (Connection conn = getConnection()) { DatabaseMetaData meta = conn.getMetaData(); ResultSet mrs = meta.getTables(null, null, null, new String[] { "TABLE" }); while (mrs.next()) tableNames.addItem(mrs.getString(3)); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } catch (IOException e) { JOptionPane.showMessageDialog(this, e); } JPanel buttonPanel = new JPanel(); add(buttonPanel, BorderLayout.SOUTH); previousButton = new JButton("Previous"); previousButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showPreviousRow(); } }); buttonPanel.add(previousButton); nextButton = new JButton("Next"); nextButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { showNextRow(); } }); buttonPanel.add(nextButton); deleteButton = new JButton("Delete"); deleteButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { deleteRow(); } }); buttonPanel.add(deleteButton); saveButton = new JButton("Save"); saveButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent event) { saveChanges(); } }); buttonPanel.add(saveButton); pack(); } /\*\* \* Prepares the text fields for showing a new table, and shows the first row. \* @param tableName the name of the table to display \*/ public void showTable(String tableName) { try { try (Connection conn = getConnection()) { // get result set Statement stat = conn.createStatement(); ResultSet result = stat.executeQuery("SELECT \* FROM " + tableName); // copy into cached row set RowSetFactory factory = RowSetProvider.newFactory(); crs = factory.createCachedRowSet(); crs.setTableName(tableName); crs.populate(result); } if (scrollPane != null) remove(scrollPane); dataPanel = new DataPanel(crs); scrollPane = new JScrollPane(dataPanel); add(scrollPane, BorderLayout.CENTER); validate(); showNextRow(); } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } } /\*\* \* Moves to the previous table row. \*/ public void showPreviousRow() { try { if (crs == null || crs.isFirst()) return; crs.previous(); dataPanel.showRow(crs); } catch (SQLException e) { for (Throwable t : e) t.printStackTrace(); } } /\*\* \* Moves to the next table row. \*/ public void showNextRow() { try { if (crs == null || crs.isLast()) return; crs.next(); dataPanel.showRow(crs); } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } } /\*\* \* Deletes current table row. \*/ public void deleteRow() { try { try (Connection conn = getConnection()) { conn.setAutoCommit(false); crs.deleteRow(); crs.acceptChanges(conn); if (crs.isAfterLast()) if (!crs.last()) crs = null; dataPanel.showRow(crs); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } } /\*\* \* Saves all changes. \*/ public void saveChanges() { try { try (Connection conn = getConnection()) { conn.setAutoCommit(false); dataPanel.setRow(crs); crs.acceptChanges(conn); } } catch (SQLException e) { JOptionPane.showMessageDialog(this, e); } } private void readDatabaseProperties() throws IOException { props = new Properties(); try (InputStream in = Files.newInputStream(Paths.get("database.properties"))) { props.load(in); } String drivers = props.getProperty("jdbc.drivers"); if (drivers != null) System.setProperty("jdbc.drivers", drivers); } /\*\* \* Gets a connection from the properties specified in the file database.properties. \* @return the database connection \*/ private Connection getConnection() throws SQLException { String url = props.getProperty("jdbc.url"); String username = props.getProperty("jdbc.username"); String password = props.getProperty("jdbc.password"); return DriverManager.getConnection(url, username, password); } } /\*\* \* This panel displays the contents of a result set. \*/class DataPanel extends JPanel { private java.util.List<JTextField> fields; /\*\* \* Constructs the data panel. \* @param rs the result set whose contents this panel displays \*/ public DataPanel(RowSet rs) throws SQLException { fields = new ArrayList<>(); setLayout(new GridBagLayout()); GridBagConstraints gbc = new GridBagConstraints(); gbc.gridwidth = 1; gbc.gridheight = 1; ResultSetMetaData rsmd = rs.getMetaData(); for (int i = 1; i <= rsmd.getColumnCount(); i++) { gbc.gridy = i - 1; String columnName = rsmd.getColumnLabel(i); gbc.gridx = 0; gbc.anchor = GridBagConstraints.EAST; add(new JLabel(columnName), gbc); int columnWidth = rsmd.getColumnDisplaySize(i); JTextField tb = new JTextField(columnWidth); if (!rsmd.getColumnClassName(i).equals("java.lang.String")) tb.setEditable(false); fields.add(tb); gbc.gridx = 1; gbc.anchor = GridBagConstraints.WEST; add(tb, gbc); } } /\*\* \* Shows a database row by populating all text fields with the column values. \*/ public void showRow(ResultSet rs) throws SQLException { for (int i = 1; i <= fields.size(); i++) { String field = rs == null ? "" : rs.getString(i); JTextField tb = fields.get(i - 1); tb.setText(field); } } /\*\* \* Updates changed data into the current row of the row set. \*/ public void setRow(RowSet rs) throws SQLException { for (int i = 1; i <= fields.size(); i++) { String field = rs.getString(i); JTextField tb = fields.get(i - 1); if (!field.equals(tb.getText())) rs.updateString(i, tb.getText()); } rs.updateRow(); } }

copycode.gif

项目目录结构如下：



其中database.properties的内容如下：

1 #jdbc.drivers=com.mysql.jdbc.Driver 2 jdbc.url=jdbc:mysql://localhost:3306/test3 jdbc.username=root 4 jdbc.password=1234

程序运行结果如下：

