

Fredy's SQL-Admin

Code Generator

Description

Doc. Version: 1.1.1
Last Update: 30. June 2002
Created by: Fredy Fischer
Created with: StarOffice 6

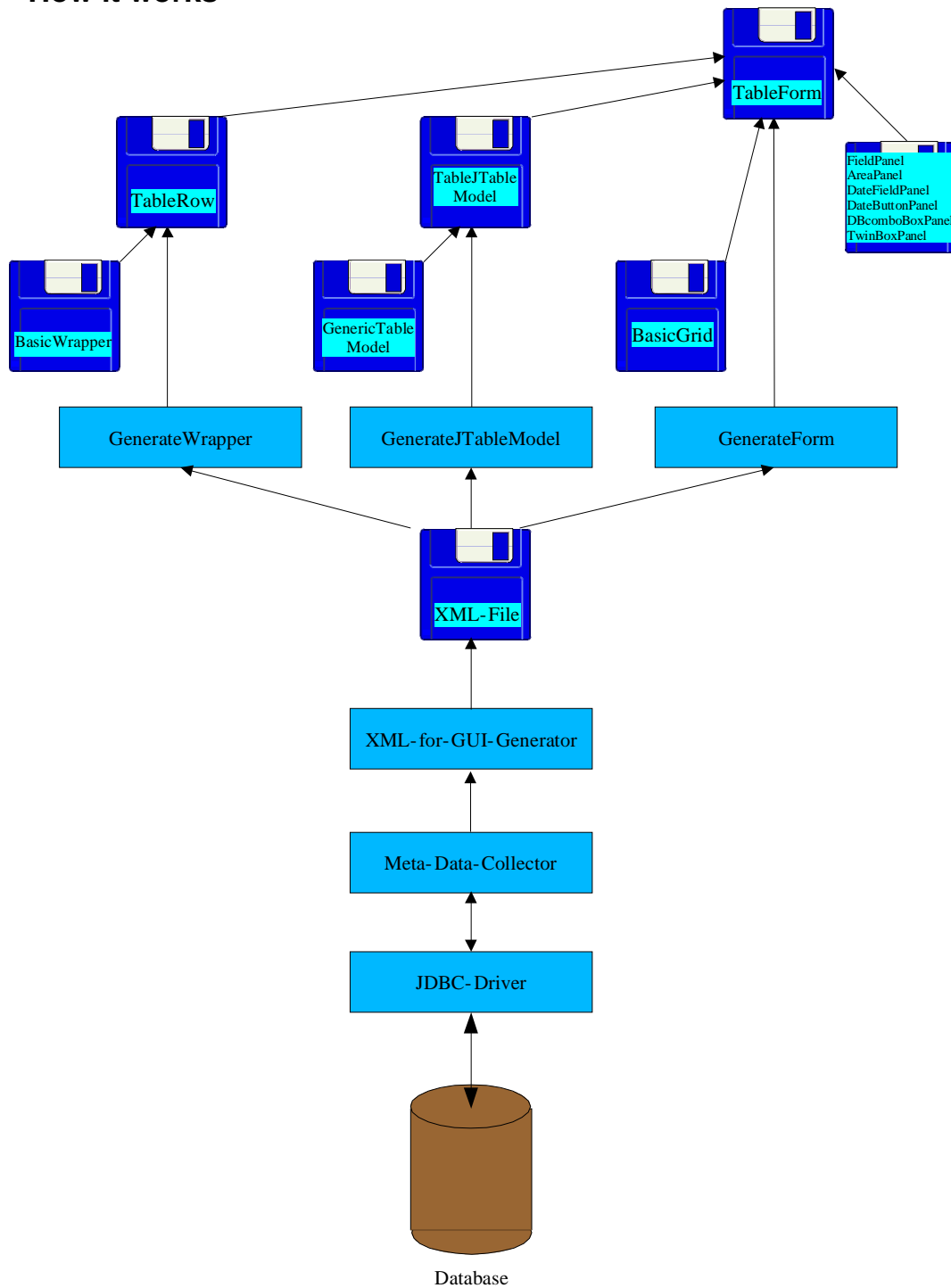
Table of Contents

1 Introduction.....	3
1.1 How it works.....	3
1.2 What is new?	4
1.3 Steps to do:.....	4
2 Requirements.....	4
3 Generator Tool.....	4
3.1 What the Parameters mean.....	5
4 GUI-Elements.....	6
5 XML-File.....	7
5.1 Structure.....	7
6 GUI-Components.....	8
6.1 Standard parameters.....	8
6.2 Label	8
6.3 FieldPanel Parameters.....	9
6.4 AreaPanel.....	9
6.5 DBcomboBoxPanel.....	10
6.6 TwinBoxPanel.....	11
6.7 DateButtonPanel.....	12
6.8 DateFieldPanel.....	13
7 Editor.....	16
7.1 How the editor looks alike.....	16
7.2 How to use the Editor.....	16
7.3 Key-Bindings.....	17
7.4 Properties Panel of a Panel.....	17
7.5 Properties of a component.....	17
8 Examples.....	19
8.1 XML-file.....	19
8.2 Swing-Form.....	20
8.3 Wrapperfile.....	28
8.4 JTableModel.....	39

1 Introduction

The Generator has been extended and offers now a higher flexibility and seems me to be a bit smarter then before.

1.1 How it works



1.2 What is new?

The Wrapperfile *TableNameRow.java* is still compatible with the older version, but it is now using PreparedStatements for all the SQL-operations, what makes it in the end much easier to handle. It now offers to update a single column by having methods like **public String updateFieldName() throws SQLException**. The generator has been changed that you need to have the primarykey defined of a table. So I was able to offer the creating of a JtableModel for each table that allows a dynamic update of a table in a JTable. This functionality has then been transferred to the new *TableNameForm* that replaces *TableNameGrid* (btw, this class is still available and fully compatible with the new *TableNameRow*).

The goal of the new generator was to generate all the files in one run. So this description is about what came out of this.

1.3 Steps to do:

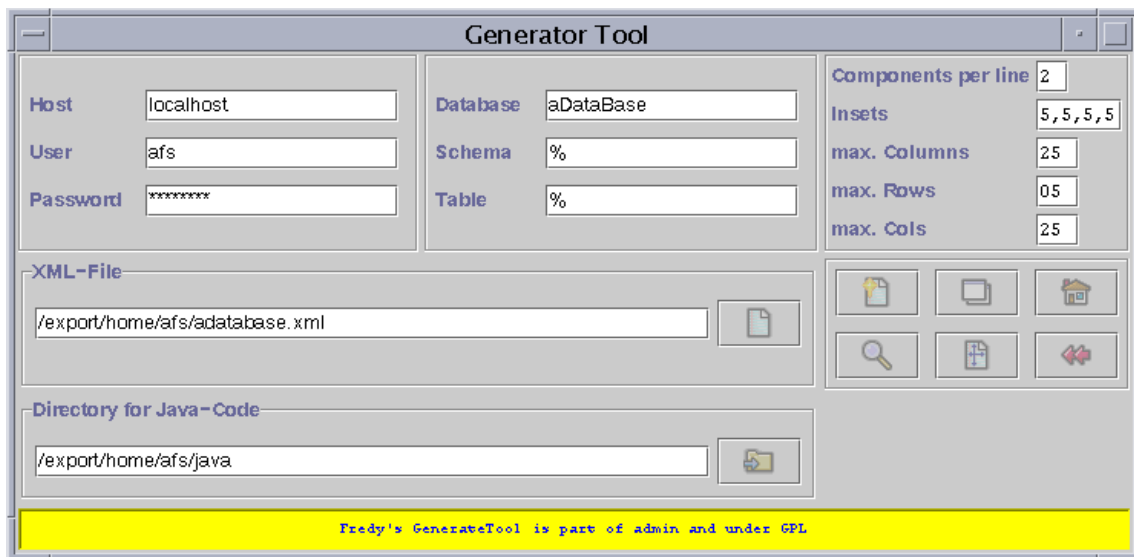
- generate the XML-File out of the DB, where you can use wildcards in the table
- edit the XML-File according your needs
- generate the code
 - this will generate the following files for each table
 - wrapperfile that makes the connection from java to the DBMS
 - the TableModel that allows to write back in every single column
 - a Swing-Form
- extend for your needs
- compile

2 Requirements

I'm using JDOM as the XML-Parser. So add it to your CLASSPATH. I got JDOM at <http://www.jdom.org>. All is done with JDK 1.3. To use the generate classes you have to have the following packages [gpl.fredy.share](#) and [gpl.fredy.ui](#) and [gpl.fredy.metadata](#) in your directories and CLASSPATH.

3 Generator Tool

The [gpl.fredy.generate.GenerateTool](#) helps you to generate the code. It generates the files mentioned aboved according standard values you put into the upper right corner of the tool.



Actually I did not start to develop the editor for the XML-file, so if you wish to steer the SWING-Form generator by yourself, so you have to continue to read ;-()

3.1 What the Parameters mean

Components per line

This value is basically set to 2, the Label and the Field. If you want to have a doubled grid, set it to 4 and so on...

Insets

These is the insets-field of the GridBagConstraints. For an explanation, please have a look at the Java-documentation (java.awt.GridBagConstraints).

MaxColumns

This is the maximum number of columns a field has before the generator does a Area to this field.

MaxRows

This indicates the mximum number of Rows a TextArea has.

MaxCols

This is the number of columns in a TextArea.

XML-File

This is the XML-file to be generated and to be generated from.

Directory for Java Code

Here you point to the upper directory under which there will be the following structure to be created: applications.[DATABASENAME].

4 GUI-Elements

There is a number of prepared GUI-Elements (gpl.fredy.ui) that are used while generating Swing-forms. This table shows the GUI-elements to be used for the different field types:

java.sql.types	Java Types internally used	Allowed GUI elements	Filter used
CHAR	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null
VARCHAR	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null
BINARY	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null
LONGVARBINARY	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null
TIME	String	FieldPanel DBcomboBoxPanel TwinBoxPanel	null
DATE	java.sql.Date	DateButtonPanel DateFieldPanel	
TIMESTAMP	String	FieldPanel DBcomboBoxPanel TwinBoxPanel	null
INTEGER	int	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.NUMERIC
NUMERIC	float	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.FLOAT
BIGINT	int	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.FLOAT
DOUBLE	double	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.FLOAT

<u>java.sql.types</u>	<i>Java Types internally used</i>	<i>Allowed GUI elements</i>	<i>Filter used</i>
FLOAT	float	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.FLOAT
BIT	int	FieldPanel DBcomboBoxPanel TwinBoxPanel	"01"
BLOB	String	FieldPanel DBcomboBoxPanel TwinBoxPanel	null
CLOB	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null
DECIMAL	float	FieldPanel DBcomboBoxPanel TwinBoxPanel	JTextFieldFilter.FLOAT
OTHER	String	FieldPanel AreaPanel DBcomboBoxPanel TwinBoxPanel	null

5 XML-File

5.1 Structure

There is in minimum one XML-File per database. This file contains rules to generate the source-file for the Swing-Form. So there will be section for each table. And each table must have one Panel, the mainPanel. This panel will be placed in the masterpanel with GridbagConstraints as shown. Under the mainPanel you can add another panel-layer. It must have the same structure as the main-Panel, mind the GridBagConstraints. As soon as the panels are done, there will be done the attributes of the table.

So this concept allows to have as many panels as you want (or fit on a screen). Make sure the section with the GridbagConstraints is done well:

```
<admin:gridBagConstraints
    insets="1,1,1,1"
    anchor="GridBagConstraints.CENTER"
    fill="GridBagConstraints.BOTH"
    weightx="1.0"
    weighty="1.0"
    gridheight="1"
```

```

        gridwidth="1"
        gridx="0"
        gridy="0"
    >
</admin:gridBagConstraints>

```

The GridBagConstraints-Element is repeated in each component to be put into the form. So while editing it, make sure, to have understand its rules.

6 GUI-Components

Each GUI-components has its specific parameters, you can change according your needs.

6.1 Standard parameters

The standard-component values are as follows:

- name is the attribute-name in the table
- type is its Java representation
- fieldType is the GUI-component used

```

<admin:component name="id"
    type="int"
    fieldType="FieldPanel"
>

```

6.2 Label

Each component has its label:

```

<admin:labelConstraints
    insets="1,1,1,1"
    anchor="GridBagConstraints.NORTHWEST"
    fill="GridBagConstraints.NONE"
    weightx="0.0"
    weighty="0.0"
    gridheight="1"
    gridwidth="1"
    gridx="0"
    gridy="0"
>
</admin:labelConstraints>

```

6.3 FieldPanel Parameters

These are the possible parameters:

- label The Label text
- length The length of the JTextField
- text The initial text of the JTextField
- filter constant from JTextFieldFilter or a string containing values
- titled true = display a titled border false no border
- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

```
<admin:parameter
    label="Identifier"
    length="11"
    text=""
    filter="JTextFieldFilter.ALPHA_NUMERIC"
    titled="true"
    title="Id"
    layout="1"
  >
</admin:parameter>
<admin:gridBagConstraints
    ....
  >
</admin:gridBagConstraints>
</admin:component>
```

6.4 AreaPanel

These are the possible parameters:

- label The Label text
- rows Number of rows
- cols Number of cols
- linewidth true or false
- text The initial text of the JTextField
- titled true = display a titled border false no border

- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

```

<admin:component name="notes"
  type="blob"
  fieldType="AreaPanel"
>
  <admin:parameter
    label="Identifier"
    rows="10"
    cols="50"
    text="Initital Text"
    lineWrap="false"
    titled="true"
    title="Notes"
    layout="1"
  >
  </admin:parameter>
<admin:gridBagConstraints
  insets="1,1,1,1"
  anchor="GridBagConstraints.NORTHEAST"
  fill="GridBagConstraints.BOTH"
  weightx="1.0"
  weighty="1.0"
  gridheight="1"
  gridwidth="1"
  gridx="0"
  gridy="1"
>
</admin:gridBagConstraints>
</admin:component>

```

6.5 DBcomboBoxPanel

This panel is a JComboBox where its values is generated out of a SQL-Query, where the first field returned from the query will be added to the JComboBox.

These are the possible parameters:

- query the SQL-query sent to the DB
- label The Label text
- text the first active value (can be null for the first)
- titled true = display a titled border false no border
- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

```
<admin:component name="notes"
  type="blob"
  fieldType="DBcomboBoxPanel"
>
  <admin:parameter
    query="select zip from locations"
    label="Identifier"
    text="a Value"
    titled="true"
    title="ZIP"
    layout="1"
  >
  </admin:parameter>
<admin:gridBagConstraints
  ...
>
</admin:gridBagConstraints>
</admin:component>
```

6.6 TwinBoxPanel

This panel is a JComboBox where its values is generated out of a SQL-Query, where the first field returned from the query will be added to the JComboBox and if selected the second value from the resultset will be given to the application.

These are the possible parameters:

- query the SQL-query sent to the DB
- label The Label text

- text the first active value (can be null for the first)
- titled true = display a titled border false no border
- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

```
<admin:component name="notes"
  type="blob"
  fieldType="TwinBoxPanel"
>
  <admin:parameter
    query="select city,zip from locations"
    label="Identifier"
    text="a Value"
    titled="true"
    title="ZIP"
    layout="1"
  >
  </admin:parameter>
<admin:gridBagConstraints
  ...
>
</admin:gridBagConstraints>
</admin:component>
```

6.7 DateButtonPanel

This is a button displaying the date as its text. If the button get pressed, a popup-window with Kai Tödter's JCalendar will be opened to offer a selection from the calendar.

These are the possible parameters:

- label The Label text
- text default Value
- titled true = display a titled border false no border
- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

```
<admin:component
    name="hiredate"
    type="java.sql.Date"
    fieldType="DateButtonPanel">
  <admin:labelConstraints
    insets="5,5,5,5"
    anchor="GridBagConstraints.NORTHWEST"
    fill="GridBagConstraints.NONE"
    weightx="0.0"
    weighty="0.0"
    gridheight="1"
    gridwidth="1"
    gridx="0" gridy="32"
  />
  <admin:parameter
    label="Hiredate"
    text=""
    titled="false"
    title=""
    layout="0"
  />
  <admin:gridBagConstraints
    insets="5,5,5,5"
    anchor="GridBagConstraints.WEST"
    fill="GridBagConstraints.NONE"
    weightx="1.0"
    weighty="1.0"
    gridheight="1"
    gridwidth="1"
    gridx="1"
    gridy="32" />
</admin:component>
```

6.8 DateFieldPanel

This is another possibility to implement a Datefield. It is based onto Martin Newstead MSeries you can get a <http://web.ukonline.co.uk/mseries/MDateSelector->

Panel.html .

These are the possible parameters:

- label The Label text
- text default Value
- titled true = display a titled border false no border
- title the title in the border
- layout 0 = standard, 1 = RAISED, 2 = LOWERED

While there are other values to set while using it, have a look at the Javadoc for this purpose.

Its representation in the XML-file:

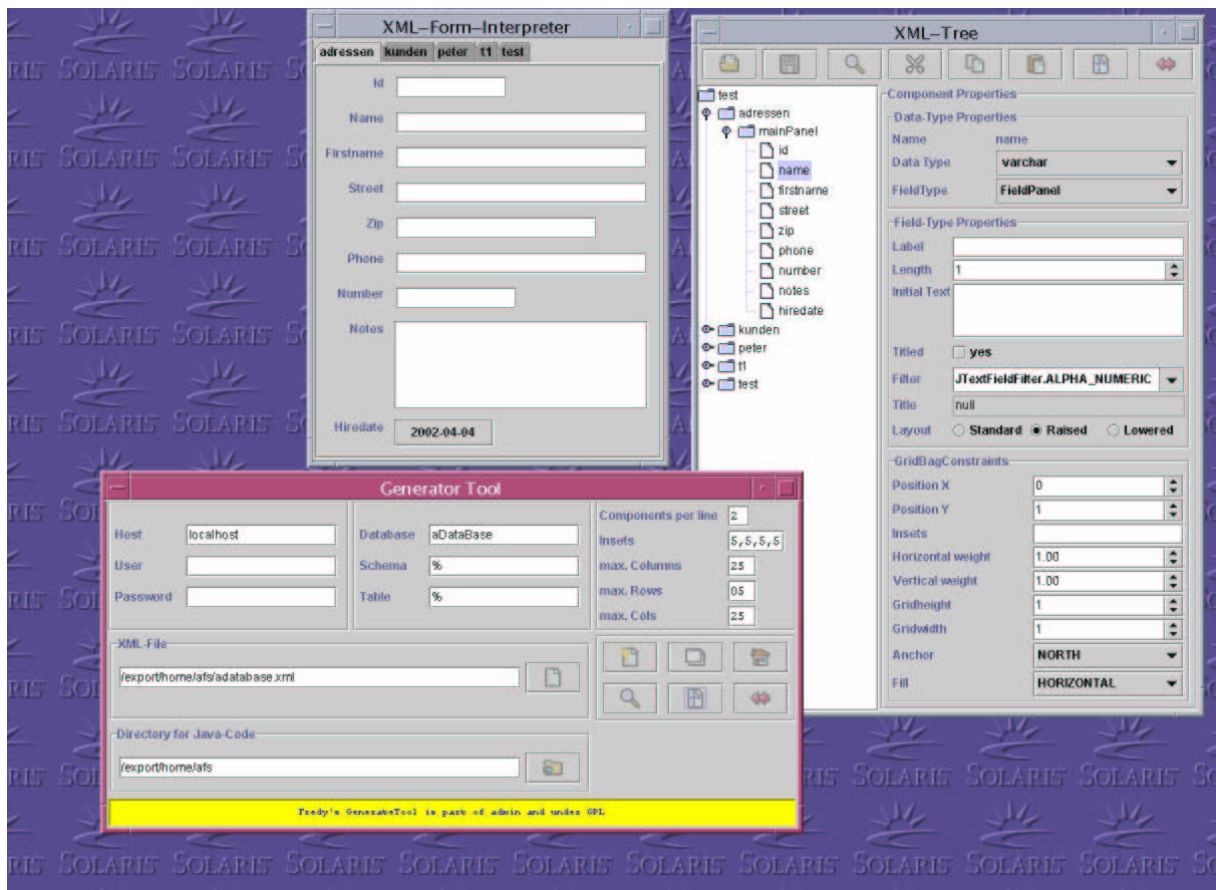
```
<admin:component
    name="hiredate"
    type="java.sql.Date"
    fieldType="DateFieldPanel">
<admin:labelConstraints
    insets="5,5,5,5"
    anchor="GridBagConstraints.NORTHWEST"
    fill="GridBagConstraints.NONE"
    weightx="0.0"
    weighty="0.0"
    gridheight="1"
    gridwidth="1"
    gridx="0" gridy="32"
/>
<admin:parameter
    label="Hiredate"
    text=""
    titled="false"
    title=""
    layout="0"
/>
<admin:gridBagConstraints
    insets="5,5,5,5"
    anchor="GridBagConstraints.WEST"
    fill="GridBagConstraints.NONE"
    weightx="1.0"
    weighty="1.0"
    gridheight="1"
```

```
        gridwidth="1"  
        gridx="1"  
        gridy="32" />  
</admin:component>
```

7 Editor

The XML-Editor is to customize the look of your Swing-GUI before you generate the code. It is not a generic XML-Editor, it has only been developed for this particular purpose. It can be started from the commandline or directly out of the generator tool.

7.1 How the editor looks alike



7.2 How to use the Editor

It first asks you to load the XML-File. It must be a XML-File as it is generated with the codegenerator. To check the content of the XML-file I actually do nothing.

As the file is correctly loaded, you will see the tree-structure of your database. This looks similar to the RDBMS-view within Admin, except there is a *Main-Panel* within each table.

A Swing-Gui needs to have the mainPanel, so you are not able to rename or delete it. You can add as many additional panels as you want (this is not really true, as I offer in maximum 500 x 500 positions to place the panel).

The best thing is, just to play around with it and make your own experience. As the values can be regenerated at any time and nothing is automatically saved,

you can play with the values, till they are fine for you. If you want to preview the form(s) you are asked to enter the connection parameters, as there are some components connecting to the database (DBComboBox and TwinBox). So you have the possibility to enter them out of the menu, or at first time you start preview, you get asked for them if they are not available. If you are starting up the editor from the Generator-Tool, they are set automatically.

7.3 Key-Bindings

Panel Menu	PopUpTrigger on the mouse or F4 (because I do not know the popupTrigger for Windows)
Properties	Properties-Key (on a Sun Keyboard, I do not know this Key on windows, so you have to click the properties-button)
Preview	F1, where if you select the DB-Tree-Node you will get a preview of all the tables where each table is in its own tab. If you select a table or a component of a table, you will see this table only.
Cut	Cut is Cut ;-)
Paste	Paste is Paste ;-)

7.4 Properties Panel of a *Panel*

The properties-panel has two sections:

- The properties section (name, title, border)
- The GridBagConstraints
these are standard [java.awt.GridBagConstraints](#), so refer to the manual for a detailed description.

7.5 Properties of a component

A component consists of a component and its label. Where a component can be one of the components described above. So you have to enter the following information to the properties:

- Label Constraints
Describes this component's label GridBagConstraints
- FieldConstraints
Describes the field GridBagConstraints
- FieldProperties
Here you enter all the information, you need for the particular GUI-component you select for this table-row. The JTextFieldFilter Component is editable and the values you type in here are used as **allowed** characters .

8 Examples

The following examples are done via the generator.

8.1 XML-file

```
<?xml version="1.0" encoding="UTF-8"?>
<database name="test" xmlns:admin="http://gotzenwil/admin">
  <!--Description of database test-->
  <admin:table name="adresses">
    <admin:primaryKeys>id</admin:primaryKeys>
    <admin:panel name="mainPanel">
      <admin:gridBagConstraints>
        <admin:insets>1,1,1,1</admin:insets>
        <admin:anchor>GridBagConstraints.CENTER</admin:anchor>
        <admin:fill>GridBagConstraints.BOTH</admin:fill>
        <admin:weightx>1.0</admin:weightx>
        <admin:weighty>1.0</admin:weighty>
        <admin:gridheight>1</admin:gridheight>
        <admin:gridwidth>1</admin:gridwidth>
        <admin:gridx>0</admin:gridx>
        <admin:gridy>0</admin:gridy>
      </admin:gridBagConstraints>
      <admin:title>Adresses</admin:title>
      <admin:border>BevelBorder(BevelBorder.LOWERED)</admin:border>
      <admin:component name="id">
        <admin:type>int</admin:type>
        <admin:fieldType>FieldPanel</admin:fieldType>
        <admin:parameter>
          <admin:label>Identifier</admin:label>
          <admin:length>11</admin:length>
          <admin:text></admin:text>
          <admin:filter>JTextFieldFilter.ALPHA_NUMERIC</admin:filter>
          <admin:titled>true</admin:titled>
          <admin:title>Id</admin:title>
          <admin:layout>1</admin:layout>
        </admin:parameter>
        <admin:gridBagConstraints>
          <admin:insets>1,1,1,1</admin:insets>
          <admin:anchor>GridBagConstraints.CENTER</admin:anchor>
          <admin:fill>GridBagConstraints.BOTH</admin:fill>
          <admin:weightx>1.0</admin:weightx>
          <admin:weighty>1.0</admin:weighty>
          <admin:gridheight>1</admin:gridheight>
          <admin:gridwidth>1</admin:gridwidth>
          <admin:gridx>0</admin:gridx>
          <admin:gridy>0</admin:gridy>
        </admin:gridBagConstraints>
      </admin:component>
      <admin:component name="notes">
        <admin:type>blob</admin:type>
        <admin:fieldType>AreaPanel</admin:fieldType>
        <admin:parameter>
          <admin:label></admin:label>
          <admin:rows>10</admin:rows>
        </admin:parameter>
      </admin:component>
    </admin:panel>
  </admin:table>
</database>
```

```

        <admin:cols>5</admin:cols>
        <admin:text>initial Text</admin:text>
        <admin:lineWrap>false</admin:lineWrap>
        <admin:titled>true</admin:titled>
        <admin:title>Notes</admin:title>
        <admin:layout>1</admin:layout>
    </admin:parameter>
    <admin:gridBagConstraints>
        <admin:insets>1,1,1,1</admin:insets>
        <admin:anchor>GridBagConstraints.CENTER</admin:anchor>
        <admin:fill>GridBagConstraints.BOTH</admin:fill>
        <admin:weightx>1.0</admin:weightx>
        <admin:weighty>1.0</admin:weighty>
        <admin:gridheight>1</admin:gridheight>
        <admin:gridwidth>1</admin:gridwidth>
        <admin:gridx>0</admin:gridx>
        <admin:gridy>1</admin:gridy>
    </admin:gridBagConstraints>
    </admin:component>
</admin:panel>
</admin:table>
</database>

```

8.2 Swing-Form

```
package applications.test;
```

```
/** this has been generated by Fredy's Admin-Tool for SQL-Databases
```

```
* Date: 2001-12-16
```

```
*
```

```
* Database: test
```

```
* Table:  adressen
```

```
*
```

```
* Admin is under GPL
```

```
*
```

```
* Fredy Fischer
```

```
* Hulmenweg 36
```

```
* 8405 Winterthur
```

```
* Switzerland
```

```
*
```

```
* se-afs@dial.eunet.ch
```

```
*
```

```
**/
```

```
import applications.basics.*;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import java.awt.datatransfer.*;
```

```
import java.util.*;
```

```
import javax.swing.*;
```

```
import javax.swing.BorderFactory;
```

```
import javax.swing.border.*;
```

```
import javax.swing.event.*;
```

```
import javax.swing.text.*;
```

```

public class AdressenForm extends BasicGrid {

// *** DECLARATIONS ***

    private AdressenTableModel tableModel;

    public FieldPanel id;
    public FieldPanel name;
    public FieldPanel firstname;
    public FieldPanel street;
    public FieldPanel zip;
    public FieldPanel phone;
    public FieldPanel number;
    public AreaPanel notes;
    public DateButtonPanel hiredate;
    AdressenRow row;

    /**
     * get the Value of the actual row as Row-Object
     * @return value of the Row-Object.
     */
    public AdressenRow getRow() {

        row = new AdressenRow(getAHost(),getAUser(),getAPassword());
        try {
            row.setId(Integer.parseInt(id.getText()));
        } catch (NumberFormatException e) { row.setId(0); }
        row.setName(name.getText());
        row.setFirstname(firstname.getText());
        row.setStreet(street.getText());
        row.setZip(zip.getText());
        row.setPhone(phone.getText());
        try {
            row.setNumber(Float.valueOf(number.getText()).floatValue());
        } catch (NumberFormatException e) { row.setNumber(0); }
        row.setNotes(notes.getText());
        try {
            row.setHiredate(java.sql.Date.valueOf(hiredate.getDate()));
        } catch (Exception e) { ; }
        return row;
    }
    /**
     * set the form with values from the Row-Object
     * @param v Value of the Row-Object.
     */
    public void setRow(AdressenRow v) {
        id.setText(Integer.toString(v.getId()));
        name.setText(v.getName());
        firstname.setText(v.getFirstname());
        street.setText(v.getStreet());
        zip.setText(v.getZip());
        phone.setText(v.getPhone());
        number.setText(Float.toString(v.getNumber()));
        notes.setText(v.getNotes());
        hiredate.setText(v.getHiredate().toString());
    }
}

```

```

/**
 * Clear the form
 */
public void clear() {
    id.clear();
    name.clear();
    firstname.clear();
    street.clear();
    zip.clear();
    phone.clear();
    number.clear();
    notes.clear();
    hiredate.clear();
}
/**
 * do the listeners for this form
 */
public void doListeners() {
    bp.clear.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            clear();
        }
    });
    bp.insert.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            row = getRow();
            msg(row.insert());
        }
    });
    bp.update.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            row = getRow();
            msg(row.update());
        }
    });
    bp.delete.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (verifyDelete() ) {
                row = getRow();
                msg(row.delete());
                clear();
            }
        }
    });
}
}

```

```
// *** MASTER PANEL ***
```

```

public JPanel formPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();

    gbc.insets = new Insets(5,5,5,5);
    gbc.anchor = GridBagConstraints.CENTER;
    gbc.fill = GridBagConstraints.BOTH;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.gridheight= 1;
    gbc.gridwidth = 1;
    gbc.gridx = 0;
}

```

```

gbc.gridy = 0;
panel.add(mainPanel(),gbc);

return panel;
}

// *** OTHER PANELS ***

private JPanel mainPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();

    gbc.insets = new Insets(5,5,5,5);
    gbc.anchor = GridBagConstraints.NORTHWEST;
    gbc.fill = GridBagConstraints.NONE;
    gbc.weightx = 0.0;
    gbc.weighty = 0.0;
    gbc.gridheight= 1;
    gbc.gridwidth = 1;
    gbc.gridx = 0;
    gbc.gridy = 0;

    panel.add(new JLabel("Id"),gbc);

    id = new FieldPanel(null,11,"",JTextFieldFilter.NUMERIC,false,"",0);

    gbc.insets = new Insets(5,5,5,5);
    gbc.anchor = GridBagConstraints.WEST;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.gridheight= 1;
    gbc.gridwidth = 1;
    gbc.gridx = 1;
    gbc.gridy = 0;

    panel.add(id,gbc);

    gbc.insets = new Insets(5,5,5,5);
    gbc.anchor = GridBagConstraints.NORTHWEST;
    gbc.fill = GridBagConstraints.NONE;
    gbc.weightx = 0.0;
    gbc.weighty = 0.0;
    gbc.gridheight= 1;
    gbc.gridwidth = 1;
    gbc.gridx = 0;
    gbc.gridy = 1;

    panel.add(new JLabel("Name"),gbc);

    name = new FieldPanel(null,25,"",null,false,"",0);

    gbc.insets = new Insets(5,5,5,5);
    gbc.anchor = GridBagConstraints.WEST;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.weightx = 1.0;

```

```

gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 1;

panel.add(name,gbc);

gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.NORTHWEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 0.0;
gbc.weighty = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 0;
gbc.gridy = 2;

panel.add(new JLabel("Firstname"),gbc);

firstname = new FieldPanel(null,25,"",null,false,"",0);

gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 2;

panel.add(firstname,gbc);

gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.NORTHWEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 0.0;
gbc.weighty = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 0;
gbc.gridy = 3;

panel.add(new JLabel("Street"),gbc);

street = new FieldPanel(null,25,"",null,false,"",0);

gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 3;

panel.add(street,gbc);

```

```
gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.NORTHWEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 0.0;
gbc.weighty = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 0;
gbc.gridy = 4;
```

```
panel.add(new JLabel("Zip"),gbc);
```

```
zip = new FieldPanel(null,20,"",null,false,"",0);
```

```
gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 4;
```

```
panel.add(zip,gbc);
```

```
gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.NORTHWEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 0.0;
gbc.weighty = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 0;
gbc.gridy = 5;
```

```
panel.add(new JLabel("Phone"),gbc);
```

```
phone = new FieldPanel(null,25,"",null,false,"",0);
```

```
gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 5;
```

```
panel.add(phone,gbc);
```

```
gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.NORTHWEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 0.0;
gbc.weighty = 0.0;
gbc.gridheight= 1;
```

```

gbc.gridwidth = 1;
gbc.gridx     = 0;
gbc.gridy     = 6;

panel.add(new JLabel("Number"),gbc);

number = new FieldPanel(null,12,"",JTextFieldFilter.FLOAT,false,"",0);

gbc.insets    = new Insets(5,5,5,5);
gbc.anchor    = GridBagConstraints.WEST;
gbc.fill      = GridBagConstraints.HORIZONTAL;
gbc.weightx   = 1.0;
gbc.weighty   = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx     = 1;
gbc.gridy     = 6;

panel.add(number,gbc);

gbc.insets    = new Insets(5,5,5,5);
gbc.anchor    = GridBagConstraints.NORTHEAST;
gbc.fill      = GridBagConstraints.NONE;
gbc.weightx   = 0.0;
gbc.weighty   = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx     = 0;
gbc.gridy     = 7;

panel.add(new JLabel("Notes"),gbc);

notes = new AreaPanel(null,5,25,true,"",false,"",0);

gbc.insets    = new Insets(5,5,5,5);
gbc.anchor    = GridBagConstraints.WEST;
gbc.fill      = GridBagConstraints.BOTH;
gbc.weightx   = 1.0;
gbc.weighty   = 1.0;
gbc.gridheight= 25;
gbc.gridwidth = 1;
gbc.gridx     = 1;
gbc.gridy     = 7;

panel.add(notes,gbc);

gbc.insets    = new Insets(5,5,5,5);
gbc.anchor    = GridBagConstraints.NORTHEAST;
gbc.fill      = GridBagConstraints.NONE;
gbc.weightx   = 0.0;
gbc.weighty   = 0.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx     = 0;
gbc.gridy     = 32;

panel.add(new JLabel("Hiredate"),gbc);

hiredate = new DateButtonPanel(null,"",false,"",0);

```

```

gbc.insets = new Insets(5,5,5,5);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.NONE;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridheight= 1;
gbc.gridwidth = 1;
gbc.gridx = 1;
gbc.gridy = 32;

panel.add(hiredate,gbc);

return panel;
}

// *** LIST PANEL ***

public JPanel listPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());
    tableModel = new AdressenTableModel(getAHost(),getAUser(),getAPassword(),getADatabase(),
getWorkingTable(),getASchema());
    theTable = new JTable(tableModel);
    panel.add(BorderLayout.CENTER,new JScrollPane(theTable));
    theTable.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
    public void valueChanged(ListSelectionEvent e) {
        if ( ! e.getValueIsAdjusting() ) {
            ListSelectionModel lsm = (ListSelectionModel)e.getSource();
            if (lsm.isSelectionEmpty()) {
                ;
            } else {
                setRow(tableModel.getRow(theTable.getSelectedRow()));
            }
        }
    }
    });
    generateQuery.search.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        tableModel.setQuery(generateQuery.getQuery());
        tableModel.executeQuery();
    }
    });
    return panel;
}

public AdressenForm(String h, String u, String p, String d, String t) {
    super(h,u,p,d,t);
}

public AdressenForm(String d, String t) {
    super(d,t);
}

public static void main(String args[]) {
    String host = null;
    String user = null;
    String password = null;
    String table = "adressen";
    String database = "test";
}

```

```

System.out.println("Syntax: java applications.test.AdressenForm -u [user] -p [password] -h [host]");
final AdressenForm form;
if (args.length < 6) {
    form = new AdressenForm(database,table);
} else {
    int i = 0;
    while ( i < args.length) {
        if (args[i].equals("-h")) {
            i++;
            host = args[i];
        }
        if (args[i].equals("-u")) {
            i++;
            user = args[i];
        }
        if (args[i].equals("-p")) {
            i++;
            password = args[i];
        }
        i++;
    }
    form = new AdressenForm(host,user,password,database,table);
}
CloseableFrame cf = new CloseableFrame("adressen");
cf.getContentPane().setLayout(new BorderLayout());
cf.getContentPane().add(form,BorderLayout.CENTER);
cf.pack();

form.bp.cancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});

cf.setVisible(true);
}

```

8.3 Wrapperfile

```

package applications.test;

/** this has been generated by Fredy's Admin-Tool for SQL-Databases
 * Date: 2001-12-16
 *
 * RDBMS:  MySQL Version: 3.23.24-beta
 * Database: test
 * Table:  adressen
 *
 * Description:
 *
 *      Primarykeys:  id
 *
 *      Columnname: id
 *      Type name : int
 *      Size      : 11
 *      Nullable  : 0
 *      Remarks   :

```

```

*      Column def: 0
*
*      Columnname: name
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: firstname
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: street
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: zip
*      Type name : varchar
*      Size      : 20
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: phone
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: number
*      Type name : float
*      Size      : 12
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: notes
*      Type name : blob
*      Size      : 65535
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: hiredate
*      Type name : date
*      Size      : 10
*      Nullable  : 1
*      Remarks   :
*      Column def:
*

```

```

* Admin is under GPL
*
* Fredy Fischer
* Hulmenweg 36
* 8405 Winterthur
* Switzerland
*
* se-afs@dial.eunet.ch
*
**/

```

```

import java.sql.*;
import java.util.Calendar;
import applications.basics.t_connect;
import applications.basics.BasicWrapper;

```

```

public class AdressenRow extends BasicWrapper {
    private ResultSet sqlresult;

    // all the prepared Statements used to treat this table
    private PreparedStatement insertStatement;
    private PreparedStatement updateStatement;
    private PreparedStatement deleteStatement;
    private PreparedStatement stmtSearchId;
    private PreparedStatement stmtUpdateId;
    private PreparedStatement stmtSearchName;
    private PreparedStatement stmtUpdateName;
    private PreparedStatement stmtSearchFirstname;
    private PreparedStatement stmtUpdateFirstname;
    private PreparedStatement stmtSearchStreet;
    private PreparedStatement stmtUpdateStreet;
    private PreparedStatement stmtSearchZip;
    private PreparedStatement stmtUpdateZip;
    private PreparedStatement stmtSearchPhone;
    private PreparedStatement stmtUpdatePhone;
    private PreparedStatement stmtSearchNumber;
    private PreparedStatement stmtUpdateNumber;
    private PreparedStatement stmtSearchNotes;
    private PreparedStatement stmtUpdateNotes;
    private PreparedStatement stmtSearchHiredate;
    private PreparedStatement stmtUpdateHiredate;

```

```

int id;

```

```

/**
 * get the value of the column id;
 * @return value of id;
 */

```

```

public int getId() { return id; }

```

```

/**
 * set the value of the column id;
 * @param v value to assign to id;
 */

```

```

public void setId(int v) { this.id = v; }

```

```

String name;

```

```
/**
 * get the value of the column name;
 * @return value of name;
 */
public String getName() { return name; }

/**
 * set the value of the column name;
 * @param v value to assign to name;
 */
public void setName(String v) { this.name = v; }
```

String firstname;

```
/**
 * get the value of the column firstname;
 * @return value of firstname;
 */
public String getFirstname() { return firstname; }

/**
 * set the value of the column firstname;
 * @param v value to assign to firstname;
 */
public void setFirstname(String v) { this.firstname = v; }
```

String street;

```
/**
 * get the value of the column street;
 * @return value of street;
 */
public String getStreet() { return street; }

/**
 * set the value of the column street;
 * @param v value to assign to street;
 */
public void setStreet(String v) { this.street = v; }
```

String zip;

```
/**
 * get the value of the column zip;
 * @return value of zip;
 */
public String getZip() { return zip; }

/**
 * set the value of the column zip;
 * @param v value to assign to zip;
 */
public void setZip(String v) { this.zip = v; }
```

String phone;

```
/**
 * get the value of the column phone;
 * @return value of phone;
 */
public String getPhone() { return phone; }
```

```
/**
 * set the value of the column phone;
 * @param v value to assign to phone;
 */
public void setPhone(String v) { this.phone = v; }
```

float number;

```
/**
 * get the value of the column number;
 * @return value of number;
 */
public float getNumber() { return number; }
```

```
/**
 * set the value of the column number;
 * @param v value to assign to number;
 */
public void setNumber(float v) { this.number = v; }
```

String notes;

```
/**
 * get the value of the column notes;
 * @return value of notes;
 */
public String getNotes() { return notes; }
```

```
/**
 * set the value of the column notes;
 * @param v value to assign to notes;
 */
public void setNotes(String v) { this.notes = v; }
```

java.sql.Date hiredate;

```
/**
 * get the value of the column hiredate;
 * @return value of hiredate;
 */
public java.sql.Date getHiredate() { return hiredate; }
```

```
/**
 * set the value of the column hiredate;
 * @param v value to assign to hiredate;
 */
public void setHiredate(java.sql.Date v) { this.hiredate = v; }
```

```

public AdressenRow(String host,
    String user,
    String password) {

    super(host,user,password,"test","adressen","%");

    this.setId(0);
    this.setName("");
    this.setFirstname("");
    this.setStreet("");
    this.setZip("");
    this.setPhone("");
    this.setNumber(0);
    this.setNotes("");
    this.setHiredate(java.sql.Date.valueOf(toDay()));
}

```

```

public AdressenRow(String host,
    String user,
    String password,
    String database,
    String table,
    String schema) {

    super(host,user,password,database,table,schema);

    this.setId(0);
    this.setName("");
    this.setFirstname("");
    this.setStreet("");
    this.setZip("");
    this.setPhone("");
    this.setNumber(0);
    this.setNotes("");
    this.setHiredate(java.sql.Date.valueOf(toDay()));
}

```

```

public AdressenRow(String host,
    String user,
    String password,
    String database,
    String table,
    String schema,
    int id,
    String name,
    String firstname,
    String street,
    String zip,
    String phone,
    float number,
    String notes,
    java.sql.Date hiredate) {
    super(host,user,password,database,table,schema);

    this.setId(id);
    this.setName(name);
}

```

```

    this.setFirstname(firstname);
    this.setStreet(street);
    this.setZip(zip);
    this.setPhone(phone);
    this.setNumber(number);
    this.setNotes(notes);
    this.setHiredate(hiredate);
}

```

```

// this section creates the prepared Statements
// for the DB-operations

```

```

public void initPreparedStatements() {

```

```

    try {

```

```

        insertStatement = con.con.prepareStatement("insert into adressen(" +
            "id," +
            "name," +
            "firstname," +
            "street," +
            "zip," +
            "phone," +
            "number," +
            "notes," +
            "hiredate)" +
            " VALUES " +
            "(?, ?, ?, ?, ?, ?, ?, ?)");

```

```

        updateStatement = con.con.prepareStatement("update adressen set " +
            "id = ? ," +
            "name = ? ," +
            "firstname = ? ," +
            "street = ? ," +
            "zip = ? ," +
            "phone = ? ," +
            "number = ? ," +
            "notes = ? ," +
            "hiredate = ? " +
            " where id = ? ");

```

```

        deleteStatement = con.con.prepareStatement("delete from adressen where id = ? ");

```

```

        stmtUpdateId = con.con.prepareStatement("update adressen set id = ? where id = ? ");

```

```

        stmtUpdateName = con.con.prepareStatement("update adressen set name = ? where id = ? ");

```

```

        stmtUpdateFirstname = con.con.prepareStatement("update adressen set firstname = ? where id = ? ");

```

```

        stmtUpdateStreet = con.con.prepareStatement("update adressen set street = ? where id = ? ");

```

```

        stmtUpdateZip = con.con.prepareStatement("update adressen set zip = ? where id = ? ");

```

```

        stmtUpdatePhone = con.con.prepareStatement("update adressen set phone = ? where id = ? ");

```

```

        stmtUpdateNumber = con.con.prepareStatement("update adressen set number = ? where id = ? ");

```

```

        stmtUpdateNotes = con.con.prepareStatement("update adressen set notes = ? where id = ? ");

```

```

        stmtUpdateHiredate = con.con.prepareStatement("update adressen set hiredate = ? where id = ? ");

```

```

        stmtSearchId = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchName = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchFirstname = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchStreet = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchZip = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchPhone = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchNumber = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchNotes = con.con.prepareStatement("select * from adressen where id = ? ");

```

```

        stmtSearchHiredate = con.con.prepareStatement("select * from adressen where id = ? ");
    } catch ( SQLException sqlException) {
        System.out.println("SQL-Exception while creating prepared Statements!\n"+
sqlException.getMessage());
    }
}

/**
 * execute this Method to get the next row
 * of a resultSet after a query has been initiated
 */
public AdressenRow next() {
    AdressenRow k = new AdressenRow(super.getHost(),super.getUser(),super.getPassword());
    try {
        if ( sqlresult.next()) {
            k.setld( sqlresult.getInt("id" ) );
            k.setName( sqlresult.getString("name" ) );
            k.setFirstname( sqlresult.getString("firstname" ) );
            k.setStreet( sqlresult.getString("street" ) );
            k.setZip( sqlresult.getString("zip" ) );
            k.setPhone( sqlresult.getString("phone" ) );
            k.setNumber( sqlresult.getFloat("number" ) );
            k.setNotes( sqlresult.getString("notes" ) );
            k.setHiredate( sqlresult.getDate("hiredate" ) );
            return k;
        } else { return null; }
    } catch (Exception e) { return null; }
}

/**
 * this method inserts a row into the table
 * and returns ok if it was succesfull
 *
 */
public String insert() {
    try {
        insertStatement.setInt(1,getld());
        insertStatement.setString(2,getName());
        insertStatement.setString(3,getFirstname());
        insertStatement.setString(4,getStreet());
        insertStatement.setString(5,getZip());
        insertStatement.setString(6,getPhone());
        insertStatement.setFloat(7,getNumber());
        insertStatement.setString(8,getNotes());
        insertStatement.setDate(9,getHiredate());
        return execQuery(insertStatement);
    } catch (SQLException sqlException ) { return sqlException.getMessage(); }
}

/**
 * this method updates this row
 * and returns ok if it was succesfull
 *
 */

```

```

public String update() {
    try {
        updateStatement.setInt(1,getId());
        updateStatement.setString(2,getName());
        updateStatement.setString(3,getFirstname());
        updateStatement.setString(4,getStreet());
        updateStatement.setString(5,getZip());
        updateStatement.setString(6,getPhone());
        updateStatement.setFloat(7,getNumber());
        updateStatement.setString(8,getNotes());
        updateStatement.setDate(9,getHiredate());
        updateStatement.setInt(10,getId());
        return execQuery(updateStatement);
    } catch (SQLException sqlException ) { return sqlException.getMessage(); }
}

```

```

/**
 * this method deletes this row
 * and returns ok if it was succesfull
 *
 */
public String delete() {
    try {
        deleteStatement.setInt(1,getId());
        return execQuery(deleteStatement);
    } catch (SQLException sqlException ) { return sqlException.getMessage(); }
}

```

```

/**
 * the following methodes enables to search rows
 * by each Field.
 * Where you can select in between a
 * <b>equal to (=)</b> query or a <b>like</b> query
 * by setting the boolean to <b>>true = equal</b> or
 * <b>false = like</b>
 */

```

```

public String searchById(int v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchId.setInt(1,v);
        return selectQuery(stmtSearchId);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where id like ?");
        p.setInt(1,v);
        return selectQuery(p);
    }
}

```

```

public String searchByName(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchName.setString(1,v);
        return selectQuery(stmtSearchName);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where name like ?");
    }
}

```

```

        p.setString(1,v);
        return selectQuery(p);
    }
}

public String searchByFirstname(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchFirstname.setString(1,v);
        return selectQuery(stmtSearchFirstname);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where firstname like ?");
        p.setString(1,v);
        return selectQuery(p);
    }
}

public String searchByStreet(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchStreet.setString(1,v);
        return selectQuery(stmtSearchStreet);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where street like ?");
        p.setString(1,v);
        return selectQuery(p);
    }
}

public String searchByZip(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchZip.setString(1,v);
        return selectQuery(stmtSearchZip);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where zip like ?");
        p.setString(1,v);
        return selectQuery(p);
    }
}

public String searchByPhone(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchPhone.setString(1,v);
        return selectQuery(stmtSearchPhone);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where phone like ?");
        p.setString(1,v);
        return selectQuery(p);
    }
}

public String searchByNumber(float v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchNumber.setFloat(1,v);
        return selectQuery(stmtSearchNumber);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where number like ?");
        p.setFloat(1,v);
        return selectQuery(p);
    }
}
}

```

```

public String searchByNotes(String v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchNotes.setString(1,v);
        return selectQuery(stmtSearchNotes);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where notes like ?");
        p.setString(1,v);
        return selectQuery(p);
    }
}

```

```

public String searchByHiredate(java.sql.Date v,boolean exact) throws SQLException {
    if (exact) {
        stmtSearchHiredate.setDate(1,v);
        return selectQuery(stmtSearchHiredate);
    } else {
        PreparedStatement p = tempStatement("select * from adressen where hiredate like ?");
        p.setDate(1,v);
        return selectQuery(p);
    }
}

```

```

/**
 * the following methodes enables to update every single Field
 **/

```

```

public String updateId() throws SQLException {
    stmtUpdateId.setInt( 1, getId() );
    updateStatement.setInt(3,getId());
    return execQuery(stmtUpdateId);
}

```

```

public String updateName() throws SQLException {
    stmtUpdateName.setString( 1, getName() );
    updateStatement.setInt(3,getId());
    return execQuery(stmtUpdateName);
}

```

```

public String updateFirstname() throws SQLException {
    stmtUpdateFirstname.setString( 1, getFirstname() );
    updateStatement.setInt(3,getId());
    return execQuery(stmtUpdateFirstname);
}

```

```

public String updateStreet() throws SQLException {
    stmtUpdateStreet.setString( 1, getStreet() );
    updateStatement.setInt(3,getId());
    return execQuery(stmtUpdateStreet);
}

```

```

public String updateZip() throws SQLException {
    stmtUpdateZip.setString( 1, getZip() );
}

```

```

        updateStatement.setInt(3,getId());
        return execQuery(stmtUpdateZip);
    }

    public String updatePhone() throws SQLException {
        stmtUpdatePhone.setString( 1, getPhone() );
        updateStatement.setInt(3,getId());
        return execQuery(stmtUpdatePhone);
    }

    public String updateNumber() throws SQLException {
        stmtUpdateNumber.setFloat( 1, getNumber() );
        updateStatement.setInt(3,getId());
        return execQuery(stmtUpdateNumber);
    }

    public String updateNotes() throws SQLException {
        stmtUpdateNotes.setString( 1, getNotes() );
        updateStatement.setInt(3,getId());
        return execQuery(stmtUpdateNotes);
    }

    public String updateHiredate() throws SQLException {
        stmtUpdateHiredate.setDate( 1, getHiredate() );
        updateStatement.setInt(3,getId());
        return execQuery(stmtUpdateHiredate);
    }
}

```

8.4 JTableModel

```
package applications.test;
```

```

/** this has been generated by Fredy's Admin-Tool for SQL-Databases
 * Date: 2001-12-16
 *
 * RDBMS: MySQL
 * Database: test
 * Table: adressen
 * Description:
 *
 * Primarykeys: id
 *
 * Columnname: id
 * Type name : int
 * Size : 11
 * Nullable : 0
 * Remarks :
 * Column def: 0
 *
 * Columnname: name

```

```

*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: firstname
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: street
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: zip
*      Type name : varchar
*      Size      : 20
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: phone
*      Type name : varchar
*      Size      : 25
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: number
*      Type name : float
*      Size      : 12
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: notes
*      Type name : blob
*      Size      : 65535
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
*      Columnname: hiredate
*      Type name : date
*      Size      : 10
*      Nullable  : 1
*      Remarks   :
*      Column def:
*
* Admin is under GPL
*
* Fredy Fischer

```

```
* Hulmenweg 36
* 8405 Winterthur
* Switzerland
*
* se-afs@dial.eunet.ch
*
```

```
put your Copyright here
```

```
**/
```

```
import java.util.Vector;
import applications.basics.*;
public class AdressenTableModel extends GenericTableModel {
```

```
    public String[] columnNames = {
        "Id",
        "Name",
        "Firstname",
        "Street",
        "Zip",
        "Phone",
        "Number",
        "Notes",
        "Hiredate"
    };
```

```
    public String getColumnName(int i) { return columnNames[i]; }
    public int getColumnCount() { return columnNames.length; }
```

```
    public String[] fieldNames = {
        "id",
        "name",
        "firstname",
        "street",
        "zip",
        "phone",
        "number",
        "notes",
        "hiredate"
    };
```

```
    public static final int ID          = 0;
    public static final int NAME        = 1;
    public static final int FIRSTNAME   = 2;
    public static final int STREET      = 3;
    public static final int ZIP         = 4;
    public static final int PHONE       = 5;
    public static final int NUMBER     = 6;
    public static final int NOTES      = 7;
    public static final int HIREDATE    = 8;
```

```
// as the Primarykey(s) are known, I can write back to the DB all fields
```

```
// not beeing part of the Primarykey
```

```
    public boolean isCellEditable(int row, int col) {
        boolean b = true;
        if ( col == ID ) b = false;
        return b;
    }
}
```

```

//setting values back to the DB

public void setValueAt(Object value, int row, int col) {
    Vector rowData = (Vector)data.elementAt(row);
    rowData.setElementAt(value,col);
    data.setElementAt(rowData,row);

    String komma = getKomma(fieldNames[col]);
    String query = "update adressen set " + fieldNames[col] + " = " + komma +
rowData.elementAt(col).toString() + komma ;
    query = query + " where id = " + getValueAt(row,ID) + """;

    SqlUpdateQuery sq = new SqlUpdateQuery(getHost(),getUser(),getPassword(),getDatabase(),
query);

    fireTableCellUpdated(row, col);
}

AdressenRow row;

/**
 * get the Value of the actual row as Row-Object
 * @return value of the Row-Object.
 */
public AdressenRow getRow(int r) {

    row = new AdressenRow(getHost(),getUser(),getPassword());
    row.setId((int)((Integer)getValueAt(r,ID)).intValue());
    row.setName((String)getValueAt(r,NAME));
    row.setFirstname((String)getValueAt(r,FIRSTNAME));
    row.setStreet((String)getValueAt(r,STREET));
    row.setZip((String)getValueAt(r,ZIP));
    row.setPhone((String)getValueAt(r,PHONE));
    row.setNumber((float)((Float)getValueAt(r,NUMBER)).floatValue());
    row.setNotes((String)getValueAt(r,NOTES));
    row.setHiredate((java.sql.Date)getValueAt(r,HIREDATE));
    return row;
}

public AdressenTableModel(String host,
    String user,
    String password,
    String database,
    String table,
    String schema) {

    super(host,user,password,database,table,schema);
    setQuery("select * from adressen ");
}

```