实训 20　综合开发 2：文本编辑器

20.1　实训目的

通过采用 Netbeans 重新开发并完善实训 15 中的文本编辑器程序，训练读者掌握使用 Netbeans 进行 AWT GUI 程序的综合开发技能。

20.2　实训案例

下面介绍如何采用 Netbeans 重新开发并完善实训 15 中的文本编辑器程序。通过本实训，读者能够掌握：

（1）GUI 的菜单设计。

（2）GUI 国际化处理。

（3）GUI 对话框的使用。

（4）剪贴板的使用。

（5）TextArea 组件的使用。

（6）熟悉 Netbeans 的 GUI Builder（GUI 生成器）的使用。

下面的操作步骤在附带光盘中有 Flash 录像，请读者参考。

20.2.1　第一步：新建项目

要求项目名称为 MyEditor，不创建主类，其余同实训 19 中的创建项目步骤。

20.2.2　第二步：新建窗体 Frame

要求类名为 EditorUI，包名为 ui。

20.2.3　第三步：添加菜单栏并设置国际化

（1）单击选中 Palette AWT 中的 Menu Bar 组件。

（2）移动鼠标到 GUI 设计区，单击，添加菜单栏，菜单栏上自动添加菜单 File 和 Edit。

（3）右击 Netbeans 左下角 Inspector 中的 Form EditorUI，弹出快捷菜单，选择 Properties 命令，弹出窗体属性对话框，勾选 Automatic Internationalization 复选框，如图 20.1 所示。这时在 Projects 窗格的 ui 源代码包下面创建了 Bundle.properties 文件。

（4）注意到 Automatic Internationalization 属性下面还有两个属性：Properties Bundle File 和 Design Locale，这里采用 default language。

（5）单击 Close 按钮。这样后面菜单的 Label 会自动被国际化处理。

（6）在 GUI 设计器中，选择 File，右击，在快捷菜单中选择 Add→MenuItem 命令。这样

就添加了菜单项 menuItem1。

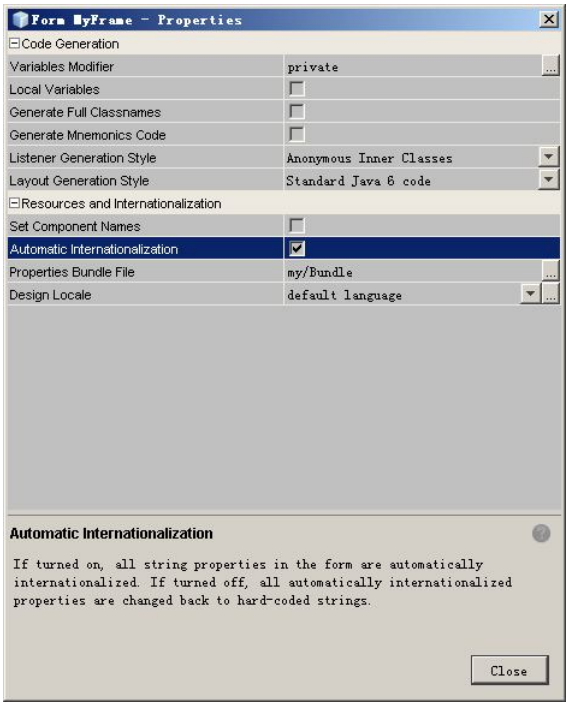

图 20.1　设置自动国际化

（7）在 Inspector 窗格中，右击，选择 Change Variable Name，弹出对话框，设置名字为 fileNew。

（8）在右边的 Properties 窗格中，修改属性 label 的值为 File。

（9）重复上面添加 fileNew 菜单项的步骤，在分别为 File 菜单添加菜单项：fileOpen（label 为 Open）、fileSave（label 为 Save）、fileSave2（label 为 Save as...）、菜单项分隔线、fileExit（label 为 Exit）。为 Edit 菜单添加菜单项： editCopy（label 为 Copy）、editPaste（label 为 Paste）。

（10）预览一下 GUI 界面，看看效果满意否。若满意则继续进行下面的步骤。

20.2.4　第四步：添加菜单项事件处理

（1）在 Inspector 窗格中选择 fileNew 菜单项，按 Enter 键或者双击，在 Netbeans 自动添加该菜单项的事件处理，并切换到事件处理的代码处，等待开发人员完善事件处理方法。

（2）找到实训 15 中计算器程序（NoteBook.java）的“新建”菜单项事件处理方法，把其代码复制过来。代码如下：

```
ta.setText("");
title = newtitle;
form.setTitle(newtitle);
path = null;
```

（3）复制过来的代码中有 ta 变量，这是 TextArea 组件变量，因此需要添加该组件。

（4）切换到 GUI 设计器窗口，从 Palette 的 AWT 中选择 Text Field 组件，并添加到 Form

的中心位置。然后修改其变量名为 ta。

（5）把 NoteBook.java 中的 title、newtitle、path 私有属性也复制过来。

（6）改上面代码中的 form 为 this。

（7）重复上面的步骤，把 File 菜单下的其他菜单项的事件处理代码都一一复制过来，并作相应的调整。

（8）加入 Edit 菜单的 editCopy 菜单项的事件处理代码，如下：

```
String text = ta.getSelectedText();
StringSelection ss = new StringSelection(text);
Toolkit.getDefaultToolkit().getSystemClipboard().setContents(ss, null);
```

（9）输入上面的代码后，Netbeans 代码编辑器的右侧会有许多红色的横线，这说明代码有错误，实际上是 StringSelection、Tool 等类所在的包没有被 import 进来。按 Ctrl+Shift+I 组合键或者执行 Source→Fix Imports 命令，这样就会自动 import 进所有需要的包。

（10）若代码不美观，则按 Alt+Shift+F 组合键或者执行 Source→Format 命令，会自动格式化代码，使其美观。

（11）重复上述操作步骤，为 editPaste 菜单项加入事件处理代码，如下：

```
Transferable t = Toolkit.getDefaultToolkit().getSystemClipboard().
             getContents(null);

try {
   if (t != null && t.isDataFlavorSupported(DataFlavor.stringFlavor)) {
      //只允许粘贴文本
      String text = (String) t.getTransferData(DataFlavor.stringFlavor);
      ta.replaceRange(text, ta.getSelectionStart(),
                    ta.getSelectionEnd());
   }
} catch (UnsupportedFlavorException e) {
} catch (IOException e) {
}
```

（12）保存所有文件。

20.2.5 第五步：运行测试

按 F6 键运行该程序，测试每个菜单项的事件处理是否正确，若正确则继续下面的步骤。

20.2.6 第六步：国际化资源

（1）在 Inspector 窗格中，在节点 Form EditorUI 上右击，选择 Properties 命令，弹出属性对话框，在最下面的 Design Locale 属性的右侧单击“...”按钮，弹出对话框，在 Language Code 文本框中选择或者输入 zh，在 Country Code 文本框中选择或者输入 CN，单击 OK 按钮，这样就创建了中文 locale。

（2）再创建英文 locale，在 Language Code 文本框中输入 en，在 Country Code 文本框中输入 US。

（3）这样就在 Projects 窗格的 ui 源包下面创建了两个新的空的文件： Bundle_zh_

CN.properties 和 Bundle_en_US.properties。

（4）右击 Bundle.properties、Bundle_zh_CN.properties 和 Bundle_en_US.properties 中的任意一个文件，从弹出的菜单中选择 Open 命令，把资源文件作对应翻译，如图 20.2 所示。

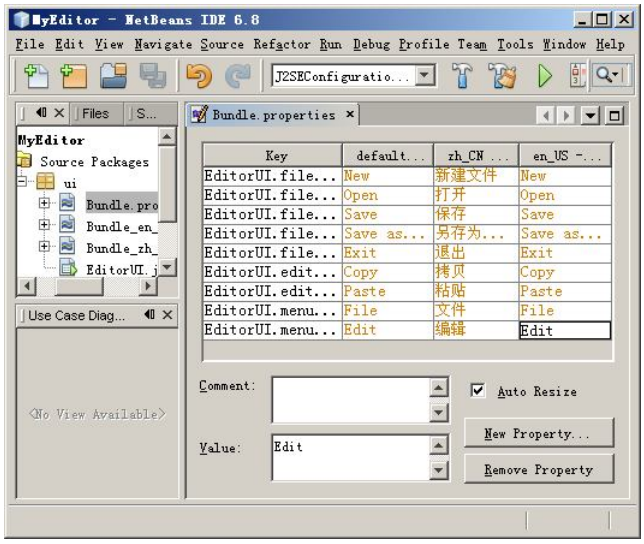

图 20.2　国际化资源翻译

（5）从操作系统的资源管理器中使用文本编辑器打开 Bundle_zh_CN.properties，会发现其中的汉字已经被转换为 Unicode 表示。

（6）在 Form EditorUI 节点的属性窗口中，选择最下面的 Design Locale 属性为 zh_CN，单击 Close 按钮。

（7）在 GUI 设计器中菜单 Label 已经变为中文，如果没有，则右击 Inspector 窗格 Form EditorUI 节点，选择 Reload Form 菜单项。

（8）单击 GUI 界面的“预览”按钮预览一下。

（9）重复上面的步骤，选择 Design Locale 为 en_US，则 GUI 设计器中的菜单 Label 变为英文。可以单击“预览”按钮预览一下。

（10）执行 Run→Set Project Configuration→Customize 命令，弹出对话框，单击左侧的 Run 按钮，在 VM Options 文框中输入-Duser.language=zh -Duser.country=CN，注意在 zh -Duser 中间有一个空格，如图 20.3 所示。单击 OK 按钮。然后按 F6 键执行程序，看看界面是否中文化。然后再设置-Duser.language=en -Duser.country=US，按 F6 键，看看界面是否英文化。

20.2.7　第七步：打包发布

执行 Run→Clean and Build Main Project 命令，则自动把当前主项目打包为 MyEditor.jar，并存放在项目中的 dist 文件夹下面。打开操作系统命令窗口，进入 dist 目录，分别执行如下命令：

```
java -jar -Duser.language=zh -Duser.country=CN MyEditor.jar
java -jar -Duser.language=en -Duser.country=US MyEditor.jar
java -jar MyEditor.jar
```

第三条命令是采用 JVM 默认的 Locale，即操作系统默认的 Locale。

上面的命令是用于执行已经发布的程序的，这样就脱离了 Netbeans IDE 开发集成环境。

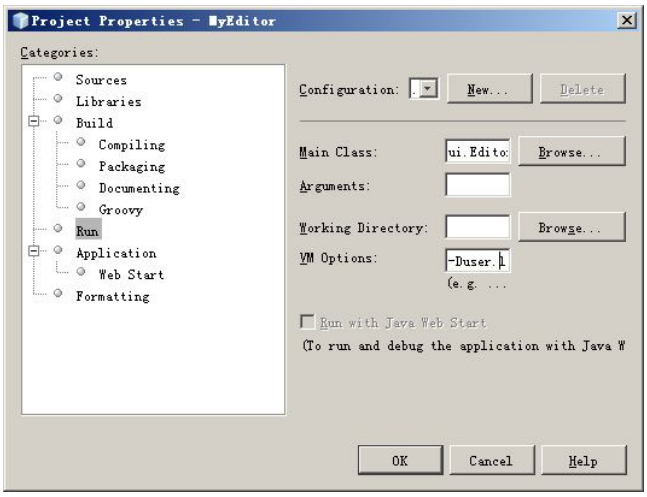

图 20.3　设置 JVM 运行参数

20.2.8　程序代码

该实训程序的全部代码如下：

```
/*
 * EditorUI.java
 *
 * Created on Dec 19, 2009, 5:16:55 PM
 */
package ui;

import java.awt.BorderLayout;
import java.awt.FileDialog;
import java.awt.Menu;
import java.awt.MenuBar;
import java.awt.MenuItem;
import java.awt.TextArea;
import java.awt.Toolkit;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.StringSelection;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
23 import java.io.BufferedReader;
24 import java.io.FileNotFoundException;
25 import java.io.FileReader;
26 import java.io.FileWriter;
27 import java.io.IOException;
28 import java.util.ResourceBundle;
29
30 /**
31  *
32  * @author Administrator
33  */
34 public class EditorUI extends java.awt.Frame {
35
36     /** Creates new form EditorUI */
37     public EditorUI() {
38         initComponents();
39     }
40
41     /** This method is called from within the constructor to
42      * initialize the form.
43      * WARNING: Do NOT modify this code. The content of this method is
44      * always regenerated by the Form Editor.
45      */
46     // <editor-fold defaultstate="collapsed" desc="Generated Code">
47     //GEN-BEGIN:initComponents
48     private void initComponents() {
49
50         ta = new TextArea();
51         menuBar1 = new MenuBar();
52         menuFile = new Menu();
53         fileNew = new MenuItem();
54         fileOpen = new MenuItem();
55         fileSave = new MenuItem();
56         fileSave2 = new MenuItem();
57         fileExit = new MenuItem();
58         menuEdit = new Menu();
59         editCopy = new MenuItem();
60         editPaste = new MenuItem();
61
62         addWindowListener(new WindowAdapter() {
63             public void windowClosing(WindowEvent evt) {
64                 exitForm(evt);
65             }
```

```
66          });
67          add(ta, BorderLayout.CENTER);
68
69          ResourceBundle bundle = ResourceBundle.getBundle("ui/Bundle");
70          menuFile.setLabel(bundle.getString("EditorUI.menuFile.label"));
71
72          fileNew.setLabel(bundle.getString("EditorUI.fileNew.label"));
73          fileNew.addActionListener(new ActionListener() {
74              public void actionPerformed(ActionEvent evt) {
75                  fileNewActionPerformed(evt);
76              }
77          });
78          menuFile.add(fileNew);
79
80          fileOpen.setLabel(bundle.getString("EditorUI.fileOpen.label"));
81          fileOpen.addActionListener(new ActionListener() {
82              public void actionPerformed(ActionEvent evt) {
83                  fileOpenActionPerformed(evt);
84              }
85          });
86          menuFile.add(fileOpen);
87
88          fileSave.setLabel(bundle.getString("EditorUI.fileSave.label"));
89          fileSave.addActionListener(new ActionListener() {
90              public void actionPerformed(ActionEvent evt) {
91                  fileSaveActionPerformed(evt);
92              }
93          });
94          menuFile.add(fileSave);
95
96          fileSave2.setLabel(bundle.getString("EditorUI.fileSave2.label"));
97          fileSave2.addActionListener(new ActionListener() {
98              public void actionPerformed(ActionEvent evt) {
99                  fileSave2ActionPerformed(evt);
100             }
101         });
102         menuFile.add(fileSave2);
103         menuFile.addSeparator();
104         fileExit.setLabel(bundle.getString("EditorUI.fileExit.label"));
105         fileExit.addActionListener(new ActionListener() {
106             public void actionPerformed(ActionEvent evt) {
107                 fileExitActionPerformed(evt);
108             }
```

```
109         });
110         menuFile.add(fileExit);
111 
112         menuBar1.add(menuFile);
113 
114         menuEdit.setLabel(bundle.getString("EditorUI.menuEdit.label"));
115 
116         editCopy.setLabel(bundle.getString("EditorUI.editCopy.label"));
117         editCopy.addActionListener(new ActionListener() {
118             public void actionPerformed(ActionEvent evt) {
119                 editCopyActionPerformed(evt);
120             }
121         });
122         menuEdit.add(editCopy);
123 
124         editPaste.setLabel(bundle.getString("EditorUI.editPaste.label"));
125         editPaste.addActionListener(new ActionListener() {
126             public void actionPerformed(ActionEvent evt) {
127                 editPasteActionPerformed(evt);
128             }
129         });
130         menuEdit.add(editPaste);
131 
132         menuBar1.add(menuEdit);
133 
134         setMenuBar(menuBar1);
135 
136         pack();
137     }// </editor-fold>//GEN-END:initComponents
138 
139     /** Exit the Application */
140     private void exitForm(WindowEvent evt) {//GEN-FIRST:event_exitForm
141         System.exit(0);
142     }//GEN-LAST:event_exitForm
143 
144     private void fileNewActionPerformed(ActionEvent evt) {
145 //GEN-FIRST:event_fileNewActionPerformed
146         ta.setText("");
147         title = newtitle;
148         this.setTitle(newtitle);
149         path = null;
150     }//GEN-LAST:event_fileNewActionPerformed
151 
```

```
152     private void fileOpenActionPerformed(ActionEvent evt) {
153     //GEN-FIRST:event_fileOpenActionPerformed
154         FileDialog dialog = new FileDialog(this);
155         dialog.setMode(FileDialog.LOAD);
156         dialog.setVisible(true);
157         if (dialog.getDirectory() == null
158                 || dialog.getFile() == null) {
159             return;
160         }
161         path = dialog.getDirectory() + "/" + dialog.getFile();
162         BufferedReader r = null;
163         try {
164             r = new BufferedReader(
165                     new FileReader(path));
166             String line = null;
167             StringBuilder sb = new StringBuilder();
168             while ((line = r.readLine()) != null) {
169                 sb.append(line).append("\n");
170             }
171             ta.setText(sb.toString());
172 
173             title = dialog.getFile();
174             this.setTitle(title);
175         } catch (FileNotFoundException e) {
176             e.printStackTrace();
177         } catch (IOException e) {
178             e.printStackTrace();
179         } finally {
180             if (r != null) {
181                 try {
182                     r.close();
183                 } catch (IOException e) {
184                     e.printStackTrace();
185                 }
186             }
187         }
188     }//GEN-LAST:event_fileOpenActionPerformed
189 
190     private void fileSaveActionPerformed(ActionEvent evt) {
191     //GEN-FIRST:event_fileSaveActionPerformed
192         if (path == null) {//若是新文档
193             FileDialog dialog = new FileDialog(this);
194             dialog.setMode(FileDialog.SAVE);
```

```
195             dialog.setVisible(true);
196             if (dialog.getDirectory() == null
197                     || dialog.getFile() == null) {
198                 return;
199             }
200             path = dialog.getDirectory() + "/" + dialog.getFile();
201             write(path);
202             title = dialog.getFile();
203             this.setTitle(title);
204         } else {
205             write(path);
206         }
207     }//GEN-LAST:event_fileSaveActionPerformed
208 
209     private void fileSave2ActionPerformed(ActionEvent evt) {
210     //GEN-FIRST:event_fileSave2ActionPerformed
211         FileDialog dialog = new FileDialog(this);
212         dialog.setMode(FileDialog.SAVE);
213         dialog.setVisible(true);
214         path = dialog.getDirectory() + "/" + dialog.getFile();
215         write(path);
216         title = dialog.getFile();
217         this.setTitle(title);
218     }//GEN-LAST:event_fileSave2ActionPerformed
219 
220     private void fileExitActionPerformed(ActionEvent evt) {
221     //GEN-FIRST:event_fileExitActionPerformed
222         System.exit(0);
223     }//GEN-LAST:event_fileExitActionPerformed
224 
225     private void editCopyActionPerformed(ActionEvent evt) {
226     //GEN-FIRST:event_editCopyActionPerformed
227         String text = ta.getSelectedText();
228         StringSelection ss = new StringSelection(text);
229         Toolkit.getDefaultToolkit().getSystemClipboard()
230 .setContents(ss, null);
231     }//GEN-LAST:event_editCopyActionPerformed
232 
233     private void editPasteActionPerformed(ActionEvent evt) {
234     //GEN-FIRST:event_editPasteActionPerformed
235         Transferable t = Toolkit.getDefaultToolkit()
236        .getSystemClipboard().getContents(null);
237 
```

```
238         try {
239             if (t != null && t.isDataFlavorSupported(
240                 DataFlavor.stringFlavor)) {
241                 //只允许粘贴文本
242                 String text = (String) t.getTransferData(
243                 DataFlavor.stringFlavor);
244                 ta.replaceRange(text, ta.getSelectionStart(),
245                         ta.getSelectionEnd());
246             }
247         } catch (UnsupportedFlavorException e) {
248         } catch (IOException e) {
249         }
250     }//GEN-LAST:event_editPasteActionPerformed
251 
252     private void write(String path) {
253         FileWriter out = null;
254         try {
255             out = new FileWriter(path);
256             out.write(ta.getText());
257         } catch (FileNotFoundException e) {
258         } catch (IOException e) {
259         } finally {
260             if (out != null) {
261                 try {
262                     out.close();
263                 } catch (IOException e) {
264                     e.printStackTrace();
265                 }
266             }
267         }
268     }
269 
270     /**
271      * @param args the command line arguments
272      */
273     public static void main(String args[]) {
274         java.awt.EventQueue.invokeLater(new Runnable() {
275 
276             public void run() {
277                 new EditorUI().setVisible(true);
278             }
279         });
280     }
```

```
281     // Variables declaration - do not modify//GEN-BEGIN:variables
282     private MenuItem editCopy;
283     private MenuItem editPaste;
284     private MenuItem fileExit;
285     private MenuItem fileNew;
286     private MenuItem fileOpen;
287     private MenuItem fileSave;
288     private MenuItem fileSave2;
289     private MenuBar menuBar1;
290     private Menu menuEdit;
291     private Menu menuFile;
292     private TextArea ta;
293     // End of variables declaration//GEN-END:variables
294     //自定义字段
295     private String newtitle = "无标题 - MyJavaEditor";
296     private String title = newtitle;
297     private String path = null;
298 }
```