Names and NSIDs: Jason Wong jaw845, Ockert Strydom ojs546

Project: jEdit

Change Number: [2,4]

Date: 11/02/2019

**Zoom the text editor**

1. **Change Request:**

Add two menu items “zoom+” and “zoom-” to scale the editors.

1. **Concept Location:**

**Table 1. The list of all the classes visited during concept location.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **File name** | **Tool used** | **Located?** | **Comments** |
| 1 | jedit\_gui.props | grep | yes | In order to locate the file in which the create the zoom buttons, we tried using grep -r to search for previous buffer (which is also under the same drop-down View. The grep outputs pointed towards many files with comments. We then tried a variety of different combinations of the words previous and buffer. When we refined the search to prev-buffer, we found the jedit\_gui.props file. |
| 2 | actions.xml | grep | yes | To find where the buttons were used, we used the label of the variable name we found in jedit. We tried using grep -r “prev-buffer.label” which allowed us to narrow down our search to only a few files and eventually found method names being called in actions.xml. |
| 3 | jedit\_keys.props | grep |  | Using the shortcut for previous buffer, we used grep -r “C+PAGE\_UP” to find where the shortcuts were declared in order to declare one for our zoom methods. |
| 4 | EditPane.java | grep | yes | To find where the method names were being used, we used grep -nr “editPane.prevBuffer()”. This allowed us to find the file and line number to observe the use as well as where to put our method since it also edits the view. |

1. **Impact Analysis:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Class name** | **Tool used** | **Impacted?** | **Comments** |
| 1 | View menu | grep | Yes | We found where the view tool bar display outputs were by using grep. Added in “Zoom In/Zoom Out” button. |
| 2 | Actions | grep | yes | We found where the actions were being called in the View tab and added in the action for “Zoom In/Zoom Out”. |
| 3 | ZoomIn() | None | Yes | Created the method ZoomIn() to zoom into the editor. |
| 4 | ZoomOut() | None | Yes | Created the method ZoomOut() to zoom out of the editor. |

1. **Learning process:**

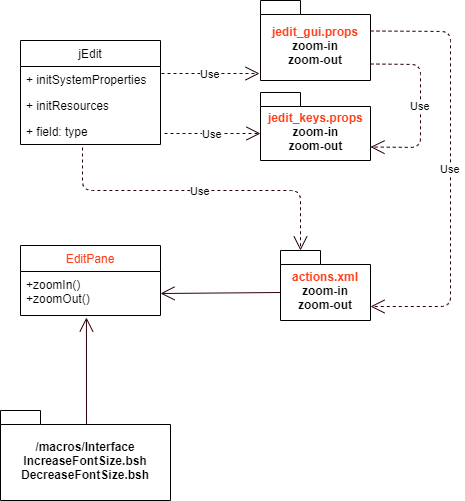
****

Figure 1. UML diagram of change 2

1. **Description of the implementation:**

jeMade changes to the jedit\_gui.props to display the “Zoom In/Zoom Out” button in the View tab. Added key shortcuts in jedit\_keys.props to assign SHIFT+UP and CONTROL+DOWN to Zoom In and Zoom Out respectively. In order to assign methodsto the buttons, we added the “Zoom In/Zoom Out” actions in the actions.xml. The buttons call the methods zoomIn() and zoomOut() in EditPane.java. These methods each call a bash script which either increases or decreases the size of the text area and gutter.

1. **Sources:**

No additional sources.

1. **Highlighted Source Code:**

**<jedit\_gui.props>**

#{{{ View menu  
view=new-view \  
 new-plain-view \  
 close-view \  
 - \  
 prev-buffer \  
 next-buffer \  
 recent-buffer \  
 show-buffer-switcher \  
 - \  
 toggle-line-numbers \  
 - \  
 zoom-in \  
 zoom-out \  
 - \  
 %scrolling \  
 %splitting \  
 %docking  
view.label=$View  
new-view.label=New $View  
new-plain-view.label=Ne$w Plain View  
close-view.label=$Close View  
prev-buffer.label=Go to $Previous Buffer  
next-buffer.label=Go to $Next Buffer  
recent-buffer.label=Go to $Recent Buffer  
show-buffer-switcher.label=Show $Buffer Switcher  
toggle-line-numbers.label=$Line Numbers  
zoom-in.label=$Zoom In  
zoom-out.label=$Zoom Out

**<jedit\_keys.props>**

prev-buffer.shortcut=C+PAGE\_UP  
next-buffer.shortcut=C+PAGE\_DOWN  
last-action.shortcut=C+SPACE  
zoom-in.shortcut=SHIFT+UP  
zoom-out.shortcut=CONTROL+DOWN  
  
recent-buffer.shortcut=C+BACK\_QUOTE  
select-block.shortcut=C+OPEN\_BRACKET

**<actions.xml>**

<ACTION NAME="prev-buffer">  
 <CODE>  
 editPane.prevBuffer();  
 </CODE>  
</ACTION>

<ACTION NAME="zoom-in">  
 <CODE>  
 editPane.zoomIn();  
 </CODE>  
</ACTION>  
  
<ACTION NAME="zoom-out">  
 <CODE>  
 editPane.zoomOut();  
 </CODE>  
</ACTION>

<ACTION NAME="toggle-line-separator">  
 <CODE>  
 buffer.toggleLineSeparator(view);  
 </CODE>  
</ACTION>

**<EditPane.java>**

//{{{ prevBuffer() method  
*/\*\*  
 \* Selects the previous buffer.  
 \** ***@since*** *jEdit 2.7pre2  
 \*/*public void prevBuffer()  
{  
 Buffer buffer = this.buffer.getPrev();  
 if(buffer == null)  
 setBuffer(jEdit.*getLastBuffer*());  
 else  
 setBuffer(buffer);  
} //}}}

*/\*\*  
 \* Zooms in by increasing the size text and gutter.  
 \*/*//{{{ zoomIn() method  
public void zoomIn(){  
 Macros.*runScript*(view, "macros/Interface/Increase\_Font\_Size.bsh",false);  
}//}}}  
  
*/\*\*  
 \* Zooms out by increasing the size text and gutter.  
 \*/*//{{{ zoomOut() method  
public void zoomOut(){  
 Macros.*runScript*(view, "macros/Interface/Decrease\_Font\_Size.bsh",false);  
}//}}}

**Add Timestamps to Log**

1. **Change Request*:***

Add timestamps to the activity log. Displays the timestamp as yyyy/MM/dd/HH/mm/ss.

1. **Concept Location:**

**Table 1. The list of all the classes visited during concept location.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **File name** | **Tool used** | **Located?** | **Comments** |
| 1 | Log.java | “Find in Files” (Ctrl-F) | No | Tried using “Find in Files” to search for logs. I couldn’t find the file, so I ended up just searching through the where the activity log was enable in the JEdit program. It was located under utilities when running so I searched in the util directory for the activity logs. |

1. **Impact Analysis:**

**Table 2. The list of all the classes visited during impact analysis.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Class name** | **Tool used** | **Impacted?** | **Comments** |
| 1 | \_log | “Find in Files” (Ctrl-F) | Yes | I found where the log outputs were by using “Find in Files” it located where the outputs of the activity logs. Added in timestamp in the \_log method. |

1. **Learning process:**

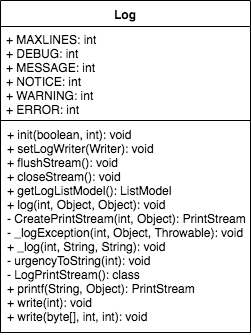
****

Figure 2. UML diagram of change 4

1. **Description of the implementation:**

Made changes to Log.java file in line 358 to 359 where the timestamp is formatted and outputted in the activity logs. It displays the year, month, day, and time. Imported java.text.\* to utilize the SimpleDateFormat method. The line of code is in the same file Log.java on line 29.

1. **Sources:**

<https://stackoverflow.com/questions/23068676/how-to-get-current-timestamp-in-string-format-in-java-yyyy-mm-dd-hh-mm-ss>

1. **Highlighted Source Code:**

**<Log.java>**

import java.io.\*;  
import java.util.\*;  
import javax.swing.\*;  
import javax.swing.event.\*;  
import java.text.\*;

…

//{{{ \_log() method  
private static void \_log(int urgency, String source, String message)  
{  
 // added timestamp here for change request 4  
 String timeStamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new Date());  
 String fullMessage = '[' + timeStamp + "]" +  
 '[' + *urgencyToString*(urgency) + "] " + source  
 + ": " + message; // added timestamp display