

Commit description

Re-show introduction  Pause

Entire word search rewritten: Now the search is done as usual, then before returning the result it is checked to be a whole word, if not, the next result is searched (and checked). A whole word has the following conditions:

start of word :

- beginning of the line
- starts with a symbol or space
- char before the first char is a symbol or space

end of word:

- end of line
- ends with a symbol or space
- char next to the last char is a symbol or space

Code changes

Below you find the code changes to review. The old version of the code is on the left, the new version is on the right.

To add a review remark, click on the respective line number. To delete it, click on it again and delete the remark's text. If a defect spans multiple lines, just mark one of those lines. If similar defects appear multiple times, please mark every occurrence. If you suspect something could be a defect but are not 100% sure, it's better to add a review remark.

At several of the change parts, you can show the whole changed method by clicking on "(Show more context)".

org/experiment/editor/search/SearchAndReplace.java, _replace()

```
1210     SearchMatcher matcher, int start, int end,
1211     boolean smartCaseReplace)
1212     throws Exception
1213     {
1214         int occurCount = 0;
1215
1216         boolean endOfLine = (buffer.getLineEndOffset(
1217             buffer.getLineOfOffset(end)) - 1 == end);
```

org/experiment/editor/search/SearchAndReplace.java, getSearchMatcher()

org/experiment/editor/search/SearchAndReplace.java, _replace() [\(Show more context\)](#)

```
1190     SearchMatcher matcher, int start, int end,
1191     boolean smartCaseReplace)
1192     throws Exception
1193     {
1194         String wordBreakChars =
1195             (String) buffer.getMode().getProperty("wordBreakChars");
1196         matcher.setNoWordSep(wordBreakChars);
1197         int occurCount = 0;
1198
1199         boolean endOfLine = (buffer.getLineEndOffset(
1200             buffer.getLineOfOffset(end)) - 1 == end);
```

org/experiment/editor/search/SearchAndReplace.java, getSearchMatcher()
[\(Show more context\)](#)

```

323     return null;
324
325     if (regexp)
326     {
327         Pattern re = Pattern.compile(search, -
328             PatternSearchMatcher.getFlag(ignoreCase));
329         matcher = new PatternSearchMatcher(re, ignoreCase);
330     }
331     else if(wholeWord)
332     {
333     String s = Pattern.quote(search);
334     String begin;
335     if (Character.isLetter(search.charAt(0)))
336     {
337     begin = "(?:\\b|^)";
338     }
339     else
340     {
341     begin = "(?:\\B|^)";
342     }
343     String end;
344     if (Character.isLetter(search.charAt(search.length() - 1)))
345     {
346     end = "(?:\\b|$)";
347     }
348     else
349     {
350     end = "(?:\\B|$)";
351     }
352     matcher = new PatternSearchMatcher(begin+s+end, ignoreCase);
353     }
354     else
355     matcher = new BoyerMooreSearchMatcher(search, ignoreCase);
356
357     return matcher;
358 }

```

org/experiment/editor/search/SearchAndReplace.java, find()

```

323     return null;
324
325     if (regexp)
326     {
327         Pattern re = Pattern.compile(search,
328             PatternSearchMatcher.getFlag(ignoreCase));
329         matcher = new PatternSearchMatcher(re, ignoreCase);
330     }
331     else
332     matcher = new BoyerMooreSearchMatcher(
333     search, ignoreCase, wholeWord);
334
335     return matcher;
336 }

```

org/experiment/editor/search/SearchAndReplace.java, find() [\(Show more context\)](#)

```
672     startOfLine = (buffer.getLineStartOffset(
673         buffer.getLineOfOffset(start)) == start);
674     endOfLine = true;
675 }
676 SearchMatcher.Match match = matcher.nextMatch(text,
677     startOfLine, endOfLine, firstTime, reverse);
678 if (match != null)
679 {
```

org/experiment/editor/search/SearchMatcher.java

```
649     startOfLine = (buffer.getLineStartOffset(
650         buffer.getLineOfOffset(start)) == start);
651     endOfLine = true;
652 }
653
654 String noWordSep =
655     (String) buffer.getMode().getProperty("noWordSep");
656 matcher.setNoWordSep(noWordSep);
657 SearchMatcher.Match match = matcher.nextMatch(text,
658     startOfLine, endOfLine, firstTime, reverse);
659 if (match != null)
660 {
```

org/experiment/editor/search/SearchMatcher.java, isEndWord()

```
109 private boolean isEndWord(char current, char next)
110 {
111     int currentCharType = TextUtilities.getCharType(current, noWordSep);
112     if (currentCharType != TextUtilities.WORD_CHAR)
113         return true;
114
115     int nextCharType = TextUtilities.getCharType(next, noWordSep);
116     return nextCharType == TextUtilities.WORD_CHAR;
117 }
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, setNoWordSep()

```
55 /**
56  * @param noWordSep the chars that are considered as word chars for
57  *                   this search
58  * @since version 4.5pre1
59  */
60 public void setNoWordSep(String noWordSep)
61 {
62     if (this.noWordSep == null)
63         this.noWordSep = "_";
64     else
65         this.noWordSep = noWordSep;
66 }
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, isWholeWord()

53

```
79  /**
80  * Check if the result is a whole word
81  * @param text the full text search
82  * @param start the start match
83  * @param end the end match
84  * @return true if the word is a whole word
85  */
86  protected boolean isWholeWord(CharSequence text, int start, int end)
87  {
88      char firstChar = text.charAt(start);
89      char prevChar = text.charAt(start - 1);
90      if (!isEndWord(firstChar, prevChar))
91      {
92          return false;
93      }
94
95      char lastChar = text.charAt(end - 1);
96      char nextChar = text.charAt(end);
97      if (!isEndWord(lastChar, nextChar))
98      {
99          return false;
100     }
101
102     return true;
103 }
```

Re-show introduction

org/experiment/editor/search/SearchMatcher.java

54

```
124  /**
125  * This should contains the noWordSep property of the edit mode of your
126  * buffer. It contains a list of chars that should be considered as word
127  * chars
128  */
129  protected String noWordSep;
```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java, getNoWordSep()

53

67

```

/**
 * Returns the noWordSep that should be used.
 * This is used by the HyperSearchOperationNode that
 * needs to remember this property since it can have
 * to restore it.
 * @return the noWordSep property
 */
String getNoWordSep()
{
    return noWordSep;
}

```


org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java

54

120

```

/**
 * true if this SearchMatcher search for whole words only.
 */
protected boolean wholeWord;

```

org/experiment/editor/search/SearchMatcher.java

org/experiment/editor/search/SearchMatcher.java

```

55     public static class Match
56     {
57         public int start;
58         public int end;
59         public String[] substitutions;
60     }

```

```

130     public static class Match
131     {
132         public int start;
133         public int end;
134         public String[] substitutions;
135
136         @Override
137         public String toString()
138         {
139             return "Match[" + start + ',' + end + ']';
140         }
141     }

```

org/experiment/editor/search/SearchDialog.java, updateEnabled()

org/experiment/editor/search/SearchDialog.java, updateEnabled() [\(Show more context\)](#)

```

705     searchForward.setEnabled(reverseEnabled);
706     if(!reverseEnabled)
707         searchForward.setSelected(true);
708
709     wholeWord.setEnabled(!regexp.isSelected());
710
711     filter.setEnabled(searchAllBuffers.isSelected()
712         || searchDirectory.isSelected());
713
714     boolean searchDirs = searchDirectory.isSelected();

```

```

705     searchForward.setEnabled(reverseEnabled);
706     if(!reverseEnabled)
707         searchForward.setSelected(true);
708
709     filter.setEnabled(searchAllBuffers.isSelected()
710         || searchDirectory.isSelected());
711
712     boolean searchDirs = searchDirectory.isSelected();

```

org/experiment/editor/search/HyperSearchOperationNode.java, getSearchMatcher()

```
224 public SearchMatcher getSearchMatcher()
225 {
226     return searchMatcher;
227 }
```

org/experiment/editor/search/HyperSearchOperationNode.java, getSearchMatcher()

Re-show introduction || Pause

```
226 public SearchMatcher getSearchMatcher()
227 {
228     // The searchMatcher has to remember the noWordSep property that
229     // was used because in case of HyperSearchOperationNode, the same
230     // SearchMatcher is used for several Buffers that can be of
231     // different edit modes.
232     searchMatcher.setNoWordSep(noWordSep);
233     return searchMatcher;
234 }
```

org/experiment/editor/search/HyperSearchOperationNode.java

```
46 private boolean treeViewDisplayed;
47 private final String searchString;
48 private List<DefaultMutableTreeNode> resultNodes;
49 private SearchMatcher searchMatcher;
```

org/experiment/editor/search/HyperSearchOperationNode.java

```
46 private boolean treeViewDisplayed;
47 private final String searchString;
48 private List<DefaultMutableTreeNode> resultNodes;
49 private SearchMatcher searchMatcher;
50 private String noWordSep;
```

org/experiment/editor/search/HyperSearchOperationNode.java, constructor

```
51 public HyperSearchOperationNode(
52     String searchString, SearchMatcher searchMatcher)
53 {
54     this.searchString = searchString;
55     this.searchMatcher = searchMatcher;
56 }
```

org/experiment/editor/search/HyperSearchOperationNode.java, constructor

```
52 public HyperSearchOperationNode(
53     String searchString, SearchMatcher searchMatcher)
54 {
55     this.searchString = searchString;
56     this.searchMatcher = searchMatcher;
57     noWordSep = searchMatcher.getNoWordSep();
58 }
```

org/experiment/editor/search/PatternSearchMatcher.java

org/experiment/editor/search/PatternSearchMatcher.java, constructor

53

Re-show introduction

```

53  /**
54  * Creates a new regular expression string matcher.
55  * @see java.util.regex.Pattern
56  * @param re the compiled regex
57  * @param ignoreCase <code>true</code> if you want to ignore case
58  * @param wholeWord <code>true</code> to search for whole word only
59  * @since version 4.5pre1
60  */
61  public PatternSearchMatcher(
62      Pattern re, boolean ignoreCase, boolean wholeWord)
63  {
64      this(re.pattern(), ignoreCase);
65      this.re = re;
66      this.wholeWord = wholeWord;
67  }

```

org/experiment/editor/search/PatternSearchMatcher.java, nextMatch()

org/experiment/editor/search/PatternSearchMatcher.java, nextMatch()
[\(Show more context\)](#)

```

156     int _end = match.end();
157
158     returnValue.start = _start;
159     returnValue.end = _end;
160
161     // For non-reversed searches, we break immediately
162     // to return the first match. For reversed searches,
163     // we continue until no more matches are found
164     if (!reverse || !match.find())

```

```

170     int _end = match.end();
171
172     returnValue.start = _start;
173     returnValue.end = _end;
174
175     if (wholeWord && !isWholeWord(text, _start, _end))
176     {
177         if (!match.find())
178             return null;
179     }
180
181     // For non-reversed searches, we break immediately
182     // to return the first match. For reversed searches,
183     // we continue until no more matches are found
184     if (!reverse || !match.find())

```

org/experiment/editor/search/PatternSearchMatcher.java, constructor

org/experiment/editor/search/PatternSearchMatcher.java, constructor
[\(Show more context\)](#)

```

58     * @since version 4.3pre13
59     */
60     public PatternSearchMatcher(Pattern re, boolean ignoreCase)
61     {
62         this(re.pattern(), ignoreCase);
63         this.re = re;
64     }

```

```

73     * @since version 4.3pre13
74     */
75     public PatternSearchMatcher(Pattern re, boolean ignoreCase)
76     {
77         this(re, ignoreCase, false);
78     }

```

org/experiment/editor/search/HyperSearchRequest.java, doHyperSearch()

```
240 private int doHyperSearch(Buffer buffer, int start, int end,
241     DefaultMutableTreeNode bufferNode)
242 {
243     int resultCount = 0;
244     EditorTextArea textArea = Editor.getActiveView().getTextArea();
245     int caretLine = textArea.getBuffer() == buffer ?
246         textArea.getCaretLine() : -1;
```

org/experiment/editor/search/HyperSearchRequest.java, doHyperSearch()

[\(Show more context\)](#)

Re-show introduction

Pause

```
240 private int doHyperSearch(Buffer buffer, int start, int end,
241     DefaultMutableTreeNode bufferNode)
242 {
243     String noWordSep =
244         (String) buffer.getMode().getProperty("noWordSep");
245     matcher.setNoWordSep(noWordSep);
246     int resultCount = 0;
247     EditorTextArea textArea = Editor.getActiveView().getTextArea();
248     int caretLine = textArea.getBuffer() == buffer ?
249         textArea.getCaretLine() : -1;
```

org/experiment/editor/search/BoyerMooreSearchMatcher.java

35

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```
35 /**
36  * Creates a new string literal matcher.
37  * @param pattern the search pattern
38  * @param ignoreCase <code>>true</code> if you want to ignore case
39  */
40 public BoyerMooreSearchMatcher(String pattern, boolean ignoreCase)
41 {
42     this(pattern, ignoreCase, false);
43 }
```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, nextMatch()

org/experiment/editor/search/BoyerMooreSearchMatcher.java, nextMatch()

[\(Show more context\)](#)

```

68     else
69     {
70         returnValue.start = pos;
71         returnValue.end = pos + pattern.length;
72         return returnValue;
73     }
74 }

```

Re-show introduction || Pause

```

82     else
83     {
84         returnValue.start = pos;
85         returnValue.end = pos + pattern.length;
86         int _end = returnValue.end;
87         if (wholeWord && !isWholeWord(text, returnValue.start, _end))
88         {
89             CharSequence subText = text.subSequence(_end, text.length());
90             Match match = nextMatch(subText,
91                 start, end, firstTime, reverse);
92             if (match == null)
93                 return null;
94             match.start = match.start + _end;
95             return match;
96         }
97         return returnValue;
98     }
99 }

```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```

35 /**
36  * Creates a new string literal matcher.
37  * @param pattern the search pattern
38  * @param ignoreCase <code>>true</code> if you want to ignore case
39  */
40 public BoyerMooreSearchMatcher(String pattern, boolean ignoreCase)
41 {
42     this.pattern = pattern.toCharArray();
43     if(ignoreCase)
44     {
45         for(int i = 0; i < this.pattern.length; i++)
46         {
47             this.pattern[i] = Character.toUpperCase(
48                 this.pattern[i]);
49         }
50     }
51     this.ignoreCase = ignoreCase;
52     pattern_end = this.pattern.length - 1;
53 }

```

org/experiment/editor/search/BoyerMooreSearchMatcher.java, constructor

```

45 /**
46  * Creates a new string literal matcher.
47  * @param pattern the search pattern
48  * @param ignoreCase <code>>true</code> if you want to ignore case
49  * @param wholeWord <code>>true</code> to search for whole word only
50  * @since 4.5pre1
51  */
52 public BoyerMooreSearchMatcher(String pattern, boolean ignoreCase,
53     boolean wholeWord)
54 {
55     this.pattern = pattern.toCharArray();
56     if(ignoreCase)
57     {
58         for(int i = 0; i < this.pattern.length; i++)
59         {
60             this.pattern[i] = Character.toUpperCase(
61                 this.pattern[i]);
62         }
63     }
64     this.ignoreCase = ignoreCase;
65     pattern_end = this.pattern.length - 1;
66     this.wholeWord = wholeWord;
67 }

```

We would now like to ask some questions on the change you just reviewed:

Re-show introduction

Pause

Is it possible to change the set of characters that determine the end of a word?

- No
- Yes
- I don't know

When the first match was not a whole word, the next match is searched. How is this implemented in `PatternSearchMatcher` and `BoyerMooreSearchMatcher`?

- It is implemented by iteration in both cases
- It is implemented by recursion in both cases
- It is implemented by recursion in one case and by iteration in the other
- I don't know

Which class decides which concrete matcher subclass to create?

- `SearchAndReplace`
- `SearchDialog`
- `SearchMatcherFactory`
- `SearchMatcher`
- I don't know

Continue ▶