Functional Specification

Thiago dos Santos Alves

March 1, 2006

Contents

Abstract

This document will describe the functionalities of a generic editor to be used inside the DevQt IDE. This editor should not has any language specific features and should be extensible via plugin system to any language that developpers wants.

Preface

What will be showed here is an overview not technical to what is expected from a generic editor that will be used on the new DevQt IDE. First in the chapter 1 is described all the features wanted on the editor and on the chapter 2 is described what possible extentions shuld be created to especialize the editor.

Chapter 1

Editor Features

1.1 Overview features

Besides edit a text file the basic editor should have the following features:

- Persistent selection;
- Multiline edition;
- Split view;
- Line numbers;
- Text folding;
- Line marks;
- Overview of line marks;
- Auto-close chars;
- Word navigation with Ctrl key;
- Highlight current line;

Ritch-Text is not acceptable, so, on copy and paste text, editor must remove formating from clipboard.

1.2 Persistent selection

When the editor is in the "Persistent Selection" state, user could select a text and start to type that editor will not overwrite the selected text (it will insert chars on the last cursor position instead) nor unselect it.

This is useful to mark an important part of the text that you know you will use it a little after. It is more useful when you split the editor view, you

could select a text on the beginning of the document and at the end of it you are writing something that will use that selected text, so, you can type the new text and see the needed one and if you need to copy and paste that text you don't need to move the cursor to the desired text and copy it, you just press Ctrl+C than Ctrl+V.

1.3 Multiline edition

When the editor is in the "Multiline" state, user is able to inser, delete or change any char in multiple lines simuntaneous. Imagine that you have the following text:

```
char chargeVar1;
char chargeVar2;
char chargeVar3;
...
char chargeVar124;
```

Now you discover that the type of the chargeVar[1..124] is not char, is double! How do you change this? First you think on find all "char" and replace it for "double", but if you do that your variable names will be affected. In this case you can put the editor in "Multiline" state, select all the 124 lines and just hit delete key four times (this will delete the "char" word) and than type "double".

1.4 Split view

With this feature user could split his code in at most 4 parts. This means that user could split horizontal or vertical, than each splitted part could be splitted once more in the other orientation.

1.5 Line numbers

The editor should present to user some kind of panel to show the line number while user types in editor. Note that this feature is not "line/column" show on status bar.

1.6 Text folding

User could "hide" and "unhide" any text on the editor. This feature should be controled by "Matcher" extention 2.1.

1.7 Line marks

Should be possible to user define any kind of mark to an specific line, for example, he could set a bookmark to a line or if some plugin is installed he could set a breakpoint to some line. This feature must be controled by "line marker" extention 2.3.

1.8 Overview of line marks

User culd see in a panel all marks of the active text without use any scroll bar. If he click on any mark, editor must set the cursor position to the position of the selected mark.

1.9 Auto-close chars

If a char like '{' is added to the text than the editor should insert automaticaly a '}' char after that, but if user continue to type and type a '}' char to close the actual openned char, editor must skip this insertion an position the cursor after the previous inserted '}' char.

1.10 Word navigation with '`Ctrl`' key

When user holds '`Ctrl`' pressend while he press directional '`Left`' or '`Right`' keys editor should skip the cursor to the beginning of the previous or next word, but as this is a *code* editor it is necessary that editor recognize as a beginning of a word an uppercase char.

With this, editor must recognize `MyClass` as two words: `My` and `Class`.

1.11 Highlight current line

User shuld be able to see a mark at the current line to represents the cursor's line.

Chapter 2

Extending editor

Editor could be extended or specialized throwght extentions that are nothing less then specialized objects that do specialized things on certain moments.

2.1 Match character

This extention is called every time the cursor of the editor changes. It should be used to match an especific character like an open brace and mark the respective close brace.

A matcher must provide the collapse and expand functions to editor.

There are tree messages that a matcher should emit:

Block enter When cursor enters on a specific block;

Block leave When cursor leaves the specific block;

Char matched When matcher founds the char that it is looking for;

2.2 Syntax highlighter

Should be possible to implement a syntax highlighter system throwght this extention. User only will have to implement an especific method that is called when the text of the editor changes.

2.3 Line marker

Besides the abilitie of add and remove marks from a line, this extention must privide an icon and a color to the given marks. This informations are getted by specific methods.

2.4 Text identer

This extension knows how to ident a given text as we type it. Could be created some kind of generic identator to be configurable as user wants.

2.5 Code completer

If a "Code completer" extention is defined, when user types an specific string on the editor, it is called to show a popup window with all possible completions of the string.