Das Handbuch zu KatePart

Thad McGinnis
Anne-Marie Mahfouf
Anders Lund
T.C. Hollingsworth
Christoph Cullmann
Lauri Watts
Übersetzer: Matthias Schulz

Das Handbuch zu KatePart

Inhaltsverzeichnis

1 Einleitung **8**

2 Grundsätzliches **9**

2.1 Ziehen und Ablegen (Drag and Drop) 9

2.2 Kurzbefehle 9

3 Arbeiten mit dem Editor von KatePart **12**

3.1 Überblick 12

3.2 Navigieren im Text 13

3.3 Arbeiten mit der Auswahl 13

3.3.1 Blockauswahl benutzen 14

3.3.2 Benutzen von Auswahl überschreiben 14

3.3.3 Benutzen von Durchgehende Auswahl 14

3.4 Kopieren und Einfügen von Text 15

3.5 Suchen und Ersetzen von Text 15

3.5.1 Die Leisten für Suchen und Ersetzen 15

3.5.2 Suchen von Text 16

3.5.3 Ersetzen 16

3.6 Lesezeichen benutzen 17

3.7 Automatischer Zeilenumbruch 17

3.8 Automatisches Einrücken benutzen 18

3.9 Kennzeichnung von Änderungen in Textzeilen 19

3.10 Die Textgrafik auf der Bildlaufleiste 19

4 Die Menüeinträge **21**

4.1 Das Menü Datei 21

4.2 Das Menü Bearbeiten 22

4.3 Das Menü Ansicht 25

4.4 Das Menü Lesezeichen 26

4.5 Das Menü Extras 27

4.6 Die Menüs „Einstellungen" und „Hilfe" 31

5 Weiterentwickelte Editierwerkzeuge **32**
5.1 Kommentar/Kommentar entfernen . . . 32
5.2 Die integrierte Befehlszeile im Editor . . . 32
5.2.1 Standardbefehle der Befehlszeile . . . 33
5.2.1.1 Befehle zum Einrichten des Editors . . . 33
5.2.1.2 Befehle zum Bearbeiten . . . 35
5.2.1.3 Befehle zur Bewegung im Dokument . . . 40
5.2.1.4 Befehle für die grundlegenden Editor-Funktionen. Diese hängen von der Anwendung ab, in der die Editorkomponente verwendet wird. . . . 41
5.3 Benutzen von Quelltextausblendung . . . 41

6 KatePart erweitern **43**
6.1 Einführung . . . 43
6.2 Arbeiten mit Syntaxhervorhebungen . . . 43
6.2.1 Überblick . . . 43
6.2.2 Das KatePart Syntaxhervorhebungssystem . . . 44
6.2.2.1 Wie es funktioniert . . . 44
6.2.2.2 Regeln . . . 45
6.2.2.3 Kontextstile und Schlüsselwörter . . . 45
6.2.2.4 Standardstile . . . 45
6.2.3 Die Hervorhebungsdefinition für das XML Format . . . 46
6.2.3.1 Überblick . . . 46
6.2.3.2 Die Abschnitte im Einzelnen . . . 49
6.2.3.3 Verfügbare Standardstile . . . 51
6.2.4 Hervorhebungs-Erkennungsregeln . . . 52
6.2.4.1 Die Regeln im Einzelnen: . . . 54
6.2.4.2 Tipps & Tricks . . . 58
6.3 Arbeiten mit Farbschemata . . . 59
6.3.1 Überblick . . . 59
6.3.2 Farbschemata für KSyntaxHighlighting . . . 60
6.3.3 Das JSON-Format der Farbschemata . . . 61
6.3.3.1 Überblick . . . 61
6.3.3.2 Dit JSON-Struktur . . . 62
6.3.3.3 Hauptabschnitte der JSON-Farbschemadateien . . . 62
6.3.3.4 Metadaten . . . 64
6.3.4 Farben im Detail: . . . 65
6.3.4.1 Editor-Farben . . . 65
6.3.4.2 Standardtextstile . . . 71
6.3.4.3 Benutzerdefinierte Textstile für Hervorhebungen . . . 74
6.3.5 Die GUI der Farbschemata . . . 75
6.3.5.1 Ein neues Schema erstellen . . . 76
6.3.5.2 JSON-Schemadateien importieren oder exportieren . . . 76

6.3.5.3 Bearbeitung von Farbschemata . . . 76
6.3.5.3.1 Farben . . . 76
6.3.5.3.2 Standardtextstile . . . 76
6.3.5.3.3 Textstile für Hervorhebungen . . . 77
6.3.6 Tipps & Tricks . . . 77
6.3.6.1 Kontrast von Textfarben . . . 77
6.3.6.2 Vorschläge zur Konsistenz bei der Syntaxhervorhebung . . . 77
6.4 Scripting mit JavaScript . . . 78
6.4.1 Einrückungsskripte . . . 78
6.4.1.1 Der Vorspann des Einrückungsskripts . . . 78
6.4.1.2 Der Quelltext des Einrückungsskripts . . . 79
6.4.2 Befehlszeilenskripte . . . 80
6.4.2.1 Der Vorspann des Befehlszeilenskripts . . . 80
6.4.2.2 Der Quelltext des Skripts . . . 81
6.4.2.2.1 Kurzbefehle festlegen . . . 82
6.4.3 Skript-API . . . 83
6.4.3.1 Cursor und Bereiche . . . 83
6.4.3.1.1 Der Cursor-Prototyp . . . 83
6.4.3.1.2 Der Bereich-Prototyp . . . 84
6.4.3.2 Globale Funktionen . . . 85
6.4.3.2.1 Lesen & Einfügen von Dateien . . . 85
6.4.3.2.2 Fehlersuche . . . 86
6.4.3.2.3 Übersetzung . . . 86
6.4.3.3 Die Programmschnittstelle zur Ansicht . . . 86
6.4.3.4 Die Programmschnittstelle zum Dokument . . . 88
6.4.3.5 Die Programmschnittstelle zum Editor . . . 93

7 Einrichten von KatePart **94**
7.1 Einstellungen für die Editor-Komponente . . . 94
7.1.1 Erscheinungsbild . . . 94
7.1.1.1 Schriftart . . . 94
7.1.1.2 Allgemein . . . 94
7.1.1.3 Randbereiche . . . 95
7.1.2 Farbschemata . . . 96
7.1.3 Bearbeitungseinstellungen . . . 97
7.1.3.1 Allgemein . . . 97
7.1.3.2 Textnavigation . . . 97
7.1.3.3 Einrückung . . . 98
7.1.3.4 Autovervollständigung . . . 100
7.1.3.5 Rechtschreibprüfung . . . 100

7.1.3.6 VI-Eingabemodus . . . 100
7.1.4 Öffnen/Speichern . . . 101
7.1.4.1 Allgemein . . . 101
7.1.4.2 Erweitert . . . 102
7.1.4.3 Modi & Dateitypen . . . 102
7.2 Einstellungen mit Dokumentvariablen . . . 103
7.2.1 Wie KatePart Variablen benutzt . . . 104
7.2.2 Verfügbare Variablen . . . 105
7.2.3 Zusätzliche Optionen in `.kateconfig`-Dateien . . . 107

8 Danksagungen und Lizenz **109**

9 Der VI-Eingabemodus **111**
9.1 VI-Eingabemodus . . . 111
9.1.1 Inkompatibilitäten mit Vim . . . 111
9.1.2 Wechseln der Modi . . . 112
9.1.3 Einbindung in Kate's Funktionen . . . 113
9.1.4 Unterstützte Befehle im normalen/visuellen Modus . . . 113
9.1.5 Unterstützte Richtungstasten . . . 115
9.1.6 Unterstützte Textobjekte . . . 116
9.1.7 Unterstützte Befehle im Eingabemodus . . . 117
9.1.8 Das Komma-Textobjekt . . . 117
9.1.9 Fehlende Funktionen . . . 118

A Reguläre Ausdrücke **119**
A.1 Einleitung . . . 119
A.2 Muster . . . 120
A.2.1 Steuerzeichen . . . 120
A.2.2 Zeichenklassen und Abkürzungen . . . 120
A.2.2.1 Zeichen mit speziellen Bedeutungen (Steuerzeichen) innerhalb von Zeichenklassen . . . 122
A.2.3 Alternativen: trifft zu wenn „eins von" . . . 122
A.2.4 Untermuster . . . 122
A.2.4.1 Angabe von Alternativen . . . 122
A.2.4.2 Speichern von gefundenem Text (Rückwärtsreferenzen) . . . 122
A.2.4.3 Vorwärtsgerichtete Behauptungen . . . 123
A.2.4.4 Rückwärtsgerichtete Behauptungen . . . 123
A.2.5 Zeichen mit speziellen Bedeutungen (Steuerzeichen) innerhalb von Mustern 123
A.3 Quantifizierer . . . 124
A.3.1 Gier . . . 125
A.3.2 In Beispielen . . . 125
A.4 Behauptungen . . . 125

B Index **127**

Zusammenfassung

KatePart ist eine voll ausgestattete Texteditorkomponente von KDE.

Kapitel 1

Einleitung

KatePart ist eine voll ausgestattete Texteditorkomponente, die in vielen Qt™- und KDE-Programmen verwendet wird. KatePart ist mehr als nur ein Texteditor, es ist als Editor für Programmierer gedacht und könnte mindestens als teilweise Alternative zu leistungsfähigeren Editoren betrachtet werden. Einer der wesentlichen Vorteile von KatePart ist die farbige Darstellung von Quelltext, angepasst für viele verschiedene Programmiersprachen wie: C/C++, Java™, Python, Perl, Bash, Modula 2, HTML und Ada.

KWrite ist eine einfache Texteditoranwendung, die auf KatePart aufbaut. KWrite hat eine Oberfläche für das Bearbeiten eines einzelnen Dokumentes (SDI). KWrite ist eine sehr einfache Anwendung, deswegen gibt es keine eigene Dokumentation. Wenn Sie KWrite benutzen können, dann können Sie KatePart überall benutzen.

Kapitel 2

Grundsätzliches

KWrite und viele andere Anwendung, die KatePart verwenden, sind sehr einfach zu nutzen. Keiner, der schon einen Texteditor benutzt hat, sollte Probleme damit haben.

2.1 Ziehen und Ablegen (Drag and Drop)

KatePart nutzt das Drag-and-Drop-Protokoll von KDE. Dateien können gezogen und auf KatePart abgelegt werden; von der Arbeitsoberfläche, von der Dateiverwaltung Dolphin, oder einer FTP-Seite, die in einem Dolphin-Fenster geöffnet ist.

2.2 Kurzbefehle

Viele der Tastenfunktionen (Tastenkürzel) sind einstellbar im Menü Einstellungen. In der Grundeinstellung hat KatePart die folgenden Tastenfunktionen:

Einfg	Umschaltung zwischen Einfüge- und Überschreibmodus. Im Einfügemodus werden alle Zeichen an der Cursor-Position eingefügt und alle Zeichen rechts vom Cursor nach rechts verschoben. Im Überschreibmodus werden die Zeichen rechts vom Cursor sofort durch die neu geschriebenen Zeichen ersetzt.
Pfeil links	Bewegt den Cursor ein Zeichen nach links.
Pfeil rechts	Bewegt den Cursor ein Zeichen nach rechts.
Pfeil hoch	Bewegt den Cursor um eine Zeile nach oben.
Pfeil runter	Bewegt den Cursor um eine Zeile nach unten.
Strg+E	Zur vorherigen Bearbeitungszeile gehen.
Strg+Umschalt+E	Zur nächsten Bearbeitungszeile gehen.
Alt+Umschalt+Pfeil hoch	Cursor zur vorherigen passenden Einrückung verschieben.
Alt+Umschalt+Pfeil runter	Cursor zur vorherigen passenden Einrückung verschieben.

Strg+6	Zur passenden Klammer gehen.
Bild auf	Bewegt den Cursor um eine Seite nach oben.
Bild ab	Bewegt den Cursor um eine Seite nach unten.
Pos 1	Setzt den Cursor an den Zeilenanfang.
Ende	Setzt den Cursor an das Zeilenende.
Strg+Pos 1	Zum Dokumentanfang.
Strg+Ende	Zum Dokumentende.
Strg+Pfeil hoch	Eine Zeile nach oben.
Strg+Pfeil runter	Eine Zeile nach unten.
Strg+Pfeil rechts	Word nach rechts.
Strg+Pfeil links	Wort nach links.
Strg+Umschalt+Pfeil hoch	Zeilen nach oben verschieben.
Strg+Umschalt+Pfeil runter	Zeilen nach unten verschieben.
Strg+Alt+Pfeil hoch	Markierte Zeilen nach oben kopieren.
Strg+Alt+Pfeil runter	Markierte Zeilen nach unten kopieren.
Strg+B	Lesezeichen hinzufügen.
Alt+Bild auf	Voriges Lesezeichen.
Alt+Bild ab	Nächstes Lesezeichen.
Entf	Löscht das Zeichen oder den markierten Text rechts vom Cursor.
Rücktaste	Löscht das Zeichen links vom Cursor.
Strg+Entf	Wort rechts löschen.
Strg+Rücktaste	Wort links löschen.
Strg+K	Zeile löschen.
Umschalt+Eingabe	Fügt eine neue Zeile mit allen Zeichen der aktuellen Zeile vom Zeilenanfang bis zum ersten Buchstaben oder bis zur ersten Zahl. Dies kann z. B. verwendet werden, um Kommentare in einen Quelltext einzufügen. Am Ende der aktuellen Zeile mit dem Inhalt „// ein Text// ” drücken Sie diese Kurzwahl und eine neue Zeile mit „// ” wird eingefügt. Die Kommentarzeichen am Beginn einer neuen Zeile für Kommentare müssen dann nicht mehr zusätzlich eingegeben werden.
Umschalt+Pfeil links	Markiert Text ein Zeichen nach links.
Umschalt+Pfeil rechts	Markiert Text ein Zeichen nach rechts.
Strg+F	Suchen.
F3	Weitersuchen.
Umschalt+F3	Frühere suchen.
Strg+H	Auswahl suchen.
Strg+Umschalt+H	Auswahl rückwärts suchen.
Strg+Umschalt+Pfeil rechts	Wort rechts auswählen.
Strg+Umschalt+Pfeil links	Wort links auswählen.
Umschalt+Pos 1	Bis zum Zeilenanfang auswählen.
Umschalt+Ende	Bis zum Zeilenende auswählen.
Umschalt+Pfeil hoch	Bis zur vorherigen Zeile auswählen.
Umschalt+Pfeil runter	Bis zur nächsten Zeile auswählen.
Strg+Umschalt+6	Bis zur passenden Klammer auswählen.
Strg+Umschalt+Bild auf	Bis zum oberen Rand der Ansicht auswählen.

Strg+Umschalt+Bild ab	Bis zum unteren Rand der Ansicht auswählen.
Umschalt+Bild auf	Bis zum Seitenanfang auswählen.
Umschalt+Bild ab	Bis zum Seitenende auswählen.
Strg+Umschalt+Pos 1	Bis zum Dokumentanfang auswählen.
Strg+Umschalt+Ende	Bis zum Dokumentende auswählen.
Strg+Pos 1	Alles auswählen.
Strg+Umschalt+A	Auswahl aufheben.
Strg+Umschalt+B	Blockauswahlmodus.
Strg+C / Strg+Einfg	Kopiert den markierten Text in die Zwischenablage.
Strg+D	Kommentar.
Strg+Umschalt+D	Kommentar entfernen.
Strg+G	Gehe zu Zeile ...
Strg+I	Auswahl einrücken.
Strg+Umschalt+I	Einrücken rückgängig.
Strg+J	Zeilen zusammenführen.
Strg+P	Drucken.
Strg+R	Ersetzen.
Strg+S	Führt den Befehl **Sichern** aus.
Strg+Umschalt+S	Speichern unter.
Strg+U	Großschreibung.
Strg+Umschalt+U	Kleinschreibung.
Strg+Alt+U	Großschreibung am Wortanfang.
Strg+V / Umschalt+Einfg	Inhalt der Zwischenablage in die aktuelle Zeile einfügen.
Strg+X / Umschalt+Einfg	Markierten Text löschen und in die Zwischenablage kopieren.
Strg+Z	Rückgängig.
Strg+Umschalt+Z	Wiederherstellen.
Strg+-	Schrift verkleinern.
Strg++Strg+=	Schrift vergrößern.
Strg+Umschalt+-	Oberste Ebene einklappen.
Strg+Umschalt++	Oberste Ebene ausklappen.
Strg+Leertaste	Quelltextvervollständigung aufrufen.
F5	Neu ladenReload.
F6	Symbolspalte anzeigen oder ausblenden.
F7	Auf Befehlszeile umschalten.
F9	Markierungen für Quelltextausblendungen anzeigen/ausblenden.
F10	Dynamischer Zeilenumbruch.
F11	Zeilennummern anzeigen/ausblenden.
Strg+T	Zeichen tauschen.
Strg+Umschalt+O	Automatische Rechtschreibprüfung.
Strg+Umschalt+V	Zu nächstem Eingabemodus wechseln.
Strg+8	Wort oben erneut verwenden.
Strg+9	Wort unten erneut verwenden.
Strg+Alt+#	Abkürzung ausschreiben.

Kapitel 3

Arbeiten mit dem Editor von KatePart

Anders Lund
Dominik Haumann
GUI-Übersetzung: Thomas Diehl
Deutsche Übersetzung: Matthias Schulz

3.1 Überblick

Der Editor von KatePart ist der Bearbeitungsbereich des KatePart-Fensters. Dieser Editor wird von Kate und KWrite benutzt und kann von Konqueror für das Anzeigen von Textdateien vom lokalen Computer oder dem Netzwerk benutzt werden.

Der Editor besteht aus den folgenden Bestandteilen:

Dem Editorbereich
: Das ist der Bereich, in den der Text Ihres Dokuments geladen wird.

Die Bildlaufleisten
: Die Bildlaufleisten zeigen die Position des sichtbaren Teils des Dokuments und können benutzt werden, um sich im Dokument zu bewegen. Ziehen an den Bildlaufleisten verändert nicht die Position des Cursors.
Die Bildlaufleisten werden nur bei Bedarf angezeigt.

Die Symbolspalte
: Die Symbolspalte ist ein kleines Feld an der linken Seite des Editorfensters, das kleine Symbole neben markierten Zeilen anzeigt.

Sie können Lesezeichen in sichtbaren Zeilen setzen oder entfernen, indem Sie mit der linken Maustaste neben der Zeile in die Symbolspalte klicken.

Die Anzeige der Symbolspalte wird mit **Ansicht** → **Symbolspalte anzeigen** ein- und ausgeschaltet.

Die Zeilennummernspalte
: Die Zeilennummernspalte zeigt die Zeilennummern aller sichtbaren Zeilen des Dokuments.
Die Anzeige der Zeilennummernspalte wird mit **Ansicht** → **Zeilennummern anzeigen** ein- und ausgeschaltet.

Die Quelltext-Ausblendungsspalte

Die Quelltext-Ausblendungsspalte erlaubt das Ein- und Ausblenden von Blöcken im Quelltext von Programmiersprachen. Die Festlegung von Anfang und Ende der Blöcke erledigt KatePart nach den Regeln in der Hervorhebungsdefinition für das aktuelle Dokument.

AUSSERDEM IN DIESEM KAPITEL:

- Navigieren im Text
- Arbeiten mit der Auswahl
- Kopieren und Einfügen von Text
- Suchen und Ersetzen von Text
- Lesezeichen benutzen
- Automatischer Zeilenumbruch
- Automatisches Einrücken benutzen

3.2 Navigieren im Text

Das Bewegen im Text funktioniert in KatePart genauso wie in anderen grafischen Editoren. Sie können den Cursor mit den Pfeiltasten bewegen und die Tasten **Bild auf**, **Bild ab**, **Pos 1** und **Ende** benutzen. Dies Alles funktioniert auch in Kombination mit den Tasten **Strg** und den **Umschalt**tasten. Die **Umschalt**tasten werden zum Auswählen benutzt, die **Strg**-Tasten haben verschiedene Bedeutungen bei verschiedenen Tasten:

- Mit den Tasten **Pfeil hoch** und **Pfeil runter** wird das Dokument verschoben und nicht der Cursor.
- In Verbindung mit den Tasten **Pfeil links** und **Pfeil rechts** wird mit diesen Tasten der Cursor wortweise bewegt.
- In Verbindung mit den Tasten **Bild auf** und **Bild ab** wird mit diesen Tasten der Cursor an den oberen oder unteren Bildrand bewegt.
- In Verbindung mit den Tasten **Pos 1** und **Ende** wird mit diesen Tasten der Cursor an den Anfang oder das Ende des Dokuments bewegt und nicht an den Anfang oder das Ende der Zeile.

KatePart stellt außerdem einen schnellen Weg bereit, um den Cursor auf eine zugehörige Klammer zu bewegen: Platzieren Sie den Cursor direkt neben eine Klammer und drücken Sie die Kombination **Strg**+**6**. Der Cursor wird zur zugehörigen öffnenden oder schließenden Klammer bewegt.

Sie können auch Lesezeichen benutzen, um den Cursor schnell auf vorher selbst definierte Positionen zu bewegen.

3.3 Arbeiten mit der Auswahl

Es gibt grundsätzlich zwei Wege, Text in KatePart zu markieren: mit der Maus oder mit der Tastatur.

Mit der Maus wird Text markiert, indem Sie mit der linken Maustaste auf den gewünschten Anfangspunkt klicken, die linke Maustaste gedrückt halten, den Mauszeiger an den gewünschten Endpunkt ziehen und dort die linke Maustaste loslassen. Der Text wird beim Ziehen markiert.

Doppelklicken auf ein Wort wählt dieses Wort aus.

Dreifachklicken auf eine Zeile wählt diese Zeile aus.

Wenn während des Klickens die **Umschalt**taste gedrückt ist, wird Text wie folgt ausgewählt:

- Wenn noch kein Text ausgewählt ist, wird der Text von der Text-Cursor-Position bis zur Mauszeigerposition ausgewählt.
- Wenn bereits eine Auswahl existiert, wird von dieser Auswahl diese Auswahl einschließend, bis zur Mauszeigerposition ausgewählt.

> ANMERKUNG
> Wenn Sie Text mit der Maus auswählen, wird dieser automatisch in die Zwischenablage kopiert und kann dann durch Klicken mit der mittleren Maustaste in eine beliebige Stelle eingefügt werden, auch außerhalb von KatePart in eine andere Anwendung.

Zum Auswählen von Text mit der Tastatur setzen Sie den Cursor auf die gewünschte Anfangsposition, halten die **Umschalt**-Taste gedrückt und bewegen dann den Cursor mit den Cursor-Tasten oder mit **Bild auf**, **Bild ab**, **Pos 1** und **Ende** an die Endposition. Wenn Sie beim Bewegen des Cursors die Taste **Strg** gedrückt halten, springt der Cursor wortweise in die gewünschte Richtung.

Sehen Sie auch unter Navigieren im Text weiter oben in diesem Kapitel nach.

Zum Kopieren der aktuellen Auswahl, wählen Sie **Bearbeiten** → **Kopieren** im Menü oder benutzen Sie den Tastaturkurzbefehl (standardmäßig **Strg**+**C**).

Zum Aufheben der aktuellen Auswahl wählen Sie **Bearbeiten** → **Auswahl aufheben** im Menü, benutzen Sie den Tastaturkurzbefehl (standardmäßig **Strg**+**Umschalt**+**A**) oder Klicken Sie mit der linken Maustaste irgendwo in das Editorfenster.

3.3.1 Blockauswahl benutzen

Wenn die Blockauswahl eingeschaltet ist, können Sie „senkrechte Auswahlen" im Text machen. Sie können also rechteckige Abschnitte mitten im Text auswählen, was sehr hilfreich z. B. für das Arbeiten mit Tabellen ist.

Die Blockauswahl können Sie im Menü mit **Bearbeiten** → **Blockauswahlmodus** oder mit der Taste **Strg**+**Umschalt**+**B** ein- und ausschalten.

3.3.2 Benutzen von Auswahl überschreiben

Wenn die Option „Auswahl überschreiben" eingeschaltet ist, dann wird die Auswahl bei der Eingabe von Text oder beim Einfügen von Text durch den eingegebenen oder eingefügten Text ersetzt. Ist sie ausgeschaltet, wird der neue Text an der Text-Cursor-Position eingefügt.

Die Option Auswahl überschreiben ist standardmäßig eingeschaltet.

Die Einstellung für diese Option wird auf der Seite Cursor & Auswahl im Einrichtungsdialog festgelegt.

3.3.3 Benutzen von Durchgehende Auswahl

Wenn diese Option eingeschaltet ist, dann bleibt die Auswahl erhalten, wenn Text eingegeben wird oder der Cursor bewegt wird.

Die Option Durchgehende Auswahl ist standardmäßig ausgeschaltet.

Die Einstellung für diese Option wird auf der Seite Cursor & Auswahl im Einrichtungsdialog festgelegt.

> WARNUNG
> Wenn beide Optionen; Durchgehende Auswahl und Auswahl überschreiben; eingeschaltet sind, wird die Auswahl ersetzt, wenn in der Auswahl Text eingegeben oder eingefügt wird. Außerdem wird die Auswahl aufgehoben.

3.4 Kopieren und Einfügen von Text

Zum Kopieren von Text, wählen Sie diesen aus und benutzen Sie dann **Bearbeiten** → **Kopieren** aus dem Menü. Sie können auch die Markierung mit der Maus vornehmen und das Kopieren in die Zwischenablage erfolgt automatisch.

Zum Einfügen von Text aus der Zwischenablage benutzen Sie **Bearbeiten** → **Einfügen** aus dem Menü.

Sie können auch Text, der mit der Maus ausgewählt wurde, durch Klicken mit der mittleren Maustaste auf die gewünschte Stelle einfügen.

> TIP
> Wenn Sie die KDE-Arbeitsumgebung benutzen, dann können Sie früher kopierten Text von allen Anwendungen in der Zwischenablage durch Klicken auf das KlipperSymbol in der Kontrollleiste; wiederfinden.

3.5 Suchen und Ersetzen von Text

3.5.1 Die Leisten für Suchen und Ersetzen

KatePart enthält eine Suchleiste für die inkrementelle Suche sowie eine erweiterte Suchleiste, die Suchen und Ersetzen mit einigen Extra-Optionen unterstützt.

Beide Leisten bieten die folgenden gemeinsamen Optionen:

Suchen
: Hier geben Sie den zu suchenden Text ein. Die Auswertung hängt von einigen der nachfolgend beschriebenen Optionen ab.

Groß-/Kleinschreibung beachten
: Wenn eingeschaltet, wird die Groß-/Kleinschreibung beim Suchen beachtet.

Die erweiterte Leiste für Suchen und Ersetzen bietet zusätzliche Einstellungen an:

Einfacher Text
: Findet alle Vorkommen des Suchtextes.

Ganze Wörter
: Wenn dies ausgewählt ist, wird die Suche nur dann als gefunden betrachtet, wenn eine Wortgrenze an beiden Seiten des Suchtextes steht, ein nicht alphanumerisches Zeichen - Leerzeichen, Zeilenende oder Sonderzeichen.

Escape-Sequenzen
: Wenn diese ausgewählt ist, wird der Menüeintrag **Hinzufügen** in den Kontextmenüs der Textfelder unten angezeigt, der es Ihnen ermöglicht, dem Suchkriterium Escape-Sequenzen aus einer vordefinierten Liste hinzuzufügen.

Regulärer Ausdruck
: Wenn dieses Feld angekreuzt ist, wird der Suchtext als regulärer Ausdruck ausgewertet. Ein Menüeintrag **Hinzufügen** wird in den Kontextmenüs der Textfelder unten angezeigt und es können reguläre Ausdrücke zum Suchtext von einer vordefinierten Liste hinzugefügt werden.
Im Abschnitt Reguläre Ausdrücke finden Sie weitere Informationen dazu.

Suchen nur in markiertem Text
Wenn dieses Feld angekreuzt ist, wird die Suche nur im ausgewählten Text durchgeführt.

Alle suchen
Das Klicken auf diesen Knopf hebt alle Fundstellen im Dokument hervor und zeigt die Anzahl der Fundstellen in einem kleinen Fenster an.

3.5.2 Suchen von Text

Um einen Text zu suchen, öffnen Sie die inkrementelle Suchleiste, indem Sie **Strg**+**F** drücken, oder aus dem Menü **Bearbeiten** → **Suche ...** auswählen.

Öffnet die Zusatzleiste für inkrementelle Suche im Editor-Fenster. An der linken Seite der Leiste finden Sie ein Symbol zum Schließen der Suchleiste, daneben befindet sich ein kleines Eingabefeld, in das Sie den gewünschten Suchbegriff eingeben können.

Sobald Sie die ersten Zeichen in das Textfeld eingeben, beginnt die Suche. Fundstellen werden im Text hervorgehoben und die Hintergrundfarbe des Textfeldes wird grün gefärbt. Falls keine Fundstellen gefunden werden, wird der Hintergrund rot eingefärbt.

Mit den Knöpfen  und  können Sie zur nächsten bzw. vorherigen Fundstelle im Dokument springen.

Die im Dokument gefundenen Textstellen bleiben auch dann noch hervorgehoben, wenn Sie die Suchleiste schließen. Drücken Sie die Taste **Esc**, um die Hervorhebungen auszuschalten.

Wenn  aktiviert ist, wird die Groß- und Kleinschreibung beim Suchen beachtet.

Klicken Sie auf  auf der rechten Seite der Suchleiste, um zwischen der inkrementellen und der erweiterten Suche zu wechseln.

Sie können die letzte Suche wiederholen ohne die Suchleiste erneut zu öffnen. Verwenden Sie dazu **Bearbeiten** → **Weitersuchen** (**F3**) oder **Bearbeiten** → **Frühere suchen** (**Umschalt+F3**).

3.5.3 Ersetzen

Um Text zu suchen und ersetzen, öffnen Sie die erweiterte Suche über **Bearbeiten** → **Ersetzen**, oder drücken Sie den Kurzbefehl **Strg**+**R**.

Links oben in der Leiste finden Sie ein Symbol, um die Leiste wieder zu schließen. Daneben befindet sich ein Kombinationsfeld, in das Sie den Suchbegriff eingeben müssen. Auch zuletzt verwendete Muster können gewählt werden.

Sie können das Verhalten der Suche mit den Optionen **Einfacher Text**, **Ganze Wörter**, **Escape-Sequenzen** und **Regulärer Ausdruck** im Auswahlfeld beeinflussen.

In den Modi **Escape-Sequenzen** und **Regulärer Ausdruck** wird der Knopf **Hinzufügen ...** unten im Kontextmenü verfügbar, über den Sie vordefinierte Zeichen/Ausdrücke für diese Modi aus einer Liste als Such- oder Ersetzungsmuster hinzufügen können.

Mit den Knöpfen  und  können Sie zur nächsten bzw. vorherigen Fundstelle im Dokument springen.

Geben Sie den Ersatztext in das Eingabefeld **Ersetzen** ein und drücken Sie anschließend den Knopf **Ersetzen** rechts daneben um die markierte Fundstelle entsprechend zu ersetzen, oder drücken Sie **Alle ersetzen**, um alle Fundstellen im ganzen Dokument zu ersetzen.

Sie können das Verhalten von Suchen und Ersetzen mit den verschiedenen Einstellungen unten in der Leiste ändern. Mit  werden nur Übereinstimmungen gefunden, die in Groß-/Kleinschreibung Ihrem Suchtext entsprechen. Mit  wird der Suchen/Ersetzen-Vorgang

nur innerhalb des ausgewählten Textes ausgeführt. Die Einstellung **Alle hervorheben** bewirkt, dass alle Fundstellen farbig hinterlegt werden und die Anzahl der Fundstellen in einem kleinen Fenster angezeigt wird. (Auch, wenn noch keine Ersetzung stattgefunden hat. Dies kann nützlich sein, um auf einen Blick zu sehen, an welchen Stellen im Dokument Ersetzungen erfolgen würden, wenn Sie auf **Alle ersetzen** klicken.)

Klicken Sie auf  rechts in der Suchleiste, um von der erweiterten Suche zur einfachen Suche zu wechseln.

> TIP
> Wenn Sie einen Regulären Ausdruck verwenden, um den Suchtext zu finden, können Sie Referenzen auf den gefundenen Text verwenden, um den gefundenen Text weiterzuverwenden.
> Im Abschnitt Reguläre Ausdrücke finden Sie weitere Informationen dazu.

> TIP
> Sie können die Befehle **find**, **replace** und **ifind** (Weitersuchen) der Befehlszeile benutzen.

3.6 Lesezeichen benutzen

Die Lesezeichenfunktion markiert bestimmte Zeilen, damit Sie diese einfach wiederfinden.

Sie können Lesezeichen auf zwei Arten setzen oder entfernen:

- Setzen Sie den Text-Cursor auf die Zeile und benutzen Sie **Lesezeichen** → **Lesezeichen setzen** im Menü oder den Tastaturkurzbefehl (**Strg**+**B**).
- Klicken auf den Symbolrand neben dieser Zeile.

Gesetzte Lesezeichen werden zum Menü **Lesezeichen** hinzugefügt. Die einzelnen Lesezeichen werden zu Menüeinträgen mit der Zeilennummer und den ersten Zeichen der Zeile als Name. Klicken Sie einfach auf den Menüeintrag und der Text-Cursor springt zur gewünschten Zeile.

Zum schnellen Bewegen des Cursors zwischen Lesezeichen oder zum nächsten/vorherigen Lesezeichen, benutzen Sie den Menüpunkt **Lesezeichen** → **Nächstes** (**Alt**+**Bild ab**) oder **Lesezeichen** → **Vorheriges** (**Alt**+**Bild auf**).

3.7 Automatischer Zeilenumbruch

Diese Funktion gestattet die Formatierung von Text in einem sehr einfachen Weg. Es werden Zeilenvorschübe eingefügt, sodass keine Zeile die vorgegebene Zeilenlänge überschreitet. Text ohne Leerzeichen, der länger als die Zeilenlänge ist, kann hiermit nicht formatiert werden.

Zum Ein- und Ausschalten diese Funktion dient das Ankreuzfeld **Statischer Zeilenumbruch** auf der Seite Editor im Einrichtungsdialog.

Die maximale Zeilenlänge wird im Feld **Zeilenumbruch bei** auf der Seite Bearbeitung im Einrichtungsdialog eingestellt.

Wenn diese Option eingeschaltet ist, funktioniert der automatische Zeilenumbruch wie folgt:

- Wenn Text eingegeben wird, fügt der Editor automatisch Zeilenvorschübe nach dem letzten Leerzeichen, das die Zeilenlänge noch nicht erreicht, ein.

- Wenn ein Dokument geladen wird, wird genauso verfahren, sodass im Dokument danach keine Zeile mehr existiert, die länger als die maximale Zeilenlänge ist, solange in allen Zeilen Leerzeichen existieren, die dieses erlauben.

> ANMERKUNG
> Es gibt keine Möglichkeit die Zeilenlänge dokumentenabhängig zu setzen oder ein- und auszuschalten. Dies wird in einer späteren Version von KatePart möglich werden..

3.8 Automatisches Einrücken benutzen

KatePart's Editorkomponente unterstützt verschiedene Varianten des automatischen Einrückens. Diese sind für verschiedene Textformate gedacht. Sie können im Menü **Extras** → **Einrückung** aus den vorhandenen Varianten eine auswählen. Der Modul für das automatische Einrücken stellt auch eine Funktion **Extras** → **Ausrichten** bereit, die die Einrückung der markierten oder der aktuellen Zeile neu berechnet. Damit können Sie durch Markieren des gesamten Textes und Nutzung dieser Funktion das Dokument neu ordnen lassen.

Alle Einrückungsmodi benutzen die Einstellungen für Einrückungen für das aktuelle Dokument.

> TIP
> Sie können alle Einstellungsvariablen, auch die für Einrückungen, setzen, indem Sie Dokumentvariablen und Dateitypen benutzen.

VERFÜGBARE EINRÜCKUNGSMETHODEN

Kein
: Diese Einstellung schaltet das automatische Einrücken ab.

Normal
: Diese Einstellung rückt die aktuelle Zeile genau so ein, wie die vorhergehende Zeile. Die ersten Nichtleerzeichen der beiden Zeilen stehen genau untereinander. Sie können diese Einstellung mit den Befehlen für Einrücken und Einrücken rückgängig kombinieren, um die Einrückung nach Ihrem persönlichen Geschmack einzustellen.

C-Stil
: Eine Einrückung für C und ähnliche Programmiersprachen, wie C++, C#, Java™, JavaScript usw. Diese Einrückung funktioniert nicht mit Skriptsprachen wie Perl oder PHP.

Haskell
: Eine Einrückung speziell für die Skriptsprache Haskell.

LilyPond
: Eine Einrückung speziell für die Lilypond-Schreibweise für Musik.

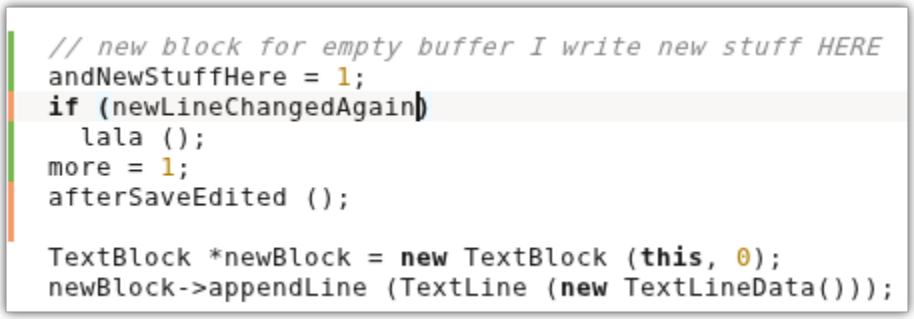
Lisp
: Eine Einrückung speziell für die Skriptsprache Lisp und deren Dialekte.

Python
: Eine Einrückung speziell für die Skriptsprache Python.

XML-Stil
: Eine Einrückung speziell für XML.

3.9 Kennzeichnung von Änderungen in Textzeilen

Mit der Kennzeichnung von Änderungen in Textzeilen in KatePart können Sie leicht erkennen, was gerade in einer Datei geändert wurde. In der Voreinstellung werden gespeicherte Änderungen werden mit grünen und noch nicht gespeicherte Änderungen mit orangefarbenen Balken links im Textfenster angezeigt.

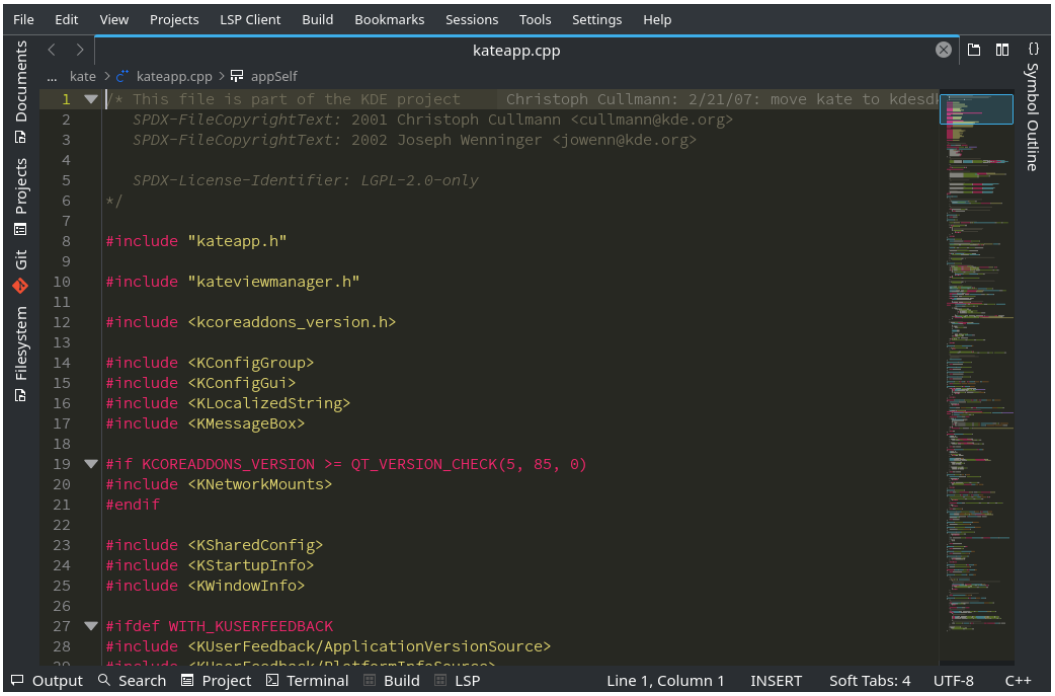


Kennzeichnung von Änderungen in Textzeilen in Aktion

Sie können die verwendeten Farben im Einrichtungsdialog auf der Seite **Schriften & Farben** ändern oder diese Funktion auf der Karteikarte **Randbereiche** auf der Seite **Erscheinungsbild** ganz abschalten.

3.10 Die Textgrafik auf der Bildlaufleiste

Die Textgrafik auf der Bildlaufleiste von KatePart zeigt eine Vorschau des Dokuments anstatt der normalen Bildlaufleiste. Der zurzeit sichtbare Teil des Dokuments ist hervorgehoben.



Die Textgrafik auf der Bildlaufleiste zeigt eine Vorschau von Kates Quelltext.

Sie können die Textgrafik auf der Bildlaufleiste temporär mit **Ansicht** → **Textgrafik auf Bildlaufleiste anzeigen** oder dauerhaft auf der Seite Erscheinungsbild im Einrichtungsdialog von KatePart ein- oder ausschalten.

Kapitel 4

Die Menüeinträge

4.1 Das Menü Datei

Datei → Neu (Strg+N)

Öffnet ein neues Dokument in einem neuen, unabhängigen Editor-Fenster.

Datei → Öffnen ... (Strg+O)

Hier erscheint ein KDE-Standarddialog zum **Datei öffnen**. Benutzen Sie das Dateifenster zum Auswählen der Datei, die Sie bearbeiten wollen und klicken Sie auf **OK** um die Datei zu öffnen.

Datei → Zuletzt geöffnete Dateien

Ist eine Abkürzung für das Öffnen der letzten bearbeiteten Dateien. Dieser Menüpunkt öffnet eine Liste mit einigen zuletzt bearbeiteten Dateien. Klicken auf eine der Dateien öffnet diese in KatePart - wenn diese Datei noch am selben Ort gespeichert ist.

Datei → Speichern (Strg+S)

Speichert die aktuelle Datei. Wenn diese noch nicht gesichert war, wird der Dialog **Speichern unter** geöffnet, ansonsten wird ohne Nachfrage überschrieben.

Datei → Speichern unter ... (Strg+Umschalt+S)

Speichert die Datei unter einem neuen Dateinamen. Die Auswahl des Dateinamens erfolgt durch einen Dialog wie bei Öffnen beschrieben.

Datei → Mit Kodierung speichern unter ...

Speichert ein Dokument unter einem neuen Dateinamen in einer anderen Kodierung.

Datei → Kopie speichern unter

Speichert eine Kopie des Dokument unter einem neuen Dateinamen, die Bearbeitung des ursprünglichen Dokuments wird fortgesetzt.

Datei → Erneut laden (F5)

Lädt die aktive Datei erneut vom Speichermedium. Dieser Befehl ist hilfreich, wenn ein anderes Programm oder ein anderer Prozess die Datei verändert hat, während diese in KatePart geöffnet war.

Datei → Drucken ... (Strg+P)

Öffnet ein einfaches Dialogfenster, in dem der Benutzer einstellen kann, was, wo und wie zu drucken ist.

Datei → Als HTML exportieren ...

Speichert das aktuelle geöffnete Dokument als HTML-Datei, die derzeitige Formatierung mit Hervorhebungen und Farbeinstellungen wird übernommen.

Datei → Schließen (Strg+W)

Dieser Befehl schließt die aktive Datei. Wenn Sie Änderungen vorgenommen haben, die noch nicht gesichert wurden, dann fragt KatePart vor dem Schließen nach, ob diese gesichert werden sollen.

Datei → Beenden (Strg+Q)

Schließt das Editorfenster. Wenn Sie jedoch mehrere KatePart-Ansichten geöffnet haben, z. B. durch **Neue** oder **Neues Fenster** werden die anderen KatePart-Ansichten nicht geschlossen.

4.2 Das Menü Bearbeiten

Bearbeiten → Rückgängig (Strg+Z)

Macht den letzten Bearbeitungsbefehl rückgängig

> ANMERKUNG
> Dieser Befehl kann eine Gruppe von gleichartigen Bearbeitungsbefehlen rückgängig machen, z. B. die Eingabe von Zeichen.

Bearbeiten → Wiederherstellen (Strg+Umschalt+Z)

Macht das letzte Rückgängig (wenn vorhanden) rückgängig.

Bearbeiten → Ausschneiden (Strg+X)

Schneidet den ausgewählten Text aus und kopiert diesen in die Zwischenablage. Die Zwischenablage funktioniert unsichtbar und ist eine Möglichkeit, Daten zwischen Anwendungen zu übertragen.

Bearbeiten → Kopieren (Strg+C)

Kopiert den ausgewählten Text in die Zwischenablage, sodass dieser an einer anderen Stelle eingefügt werden kann. Die Zwischenablage funktioniert unsichtbar und ist eine Möglichkeit, Daten zwischen Anwendungen zu übertragen.

Bearbeiten → Einfügen (Strg+V)

Fügt den ersten Eintrag in der Zwischenablage an der Cursor-Position ein. Die Zwischenablage funktioniert unsichtbar und ist eine Möglichkeit, Daten zwischen Anwendungen zu übertragen.

> ANMERKUNG
> Wenn die Option Auswahl überschreiben eingeschaltet ist, dann überschreibt der eingefügte Text eine vorhandene Auswahl.

Bearbeiten → Auswahl ausschneiden (Strg+Umschalt+Einfg)

Damit wird der vorher ausgewählte Inhalt der Mausauswahl eingefügt. Markieren Sie Text mit dem Mauszeiger und fügen sie ihn mit dieser Menüaktion in die aktuell geöffnete Datei ein.

Bearbeiten → Mit Inhalt der Zwischenablage tauschen

Damit wird der ausgewählten Text mit dem Inhalt der Zwischenablage getauscht.

Bearbeiten → Verlauf der Zwischenablage

In diesem Untermenü wird der Anfang von Texten angezeigt, die zuletzt in die Zwischenablage kopiert wurden. Wählen Sie einen dieser Einträge, um ihn in die aktuell geöffnete Datei einzufügen.

Bearbeiten → Als HTML kopieren

Kopiert die Auswahl als HTML, formatiert mit den derzeitigen Hervorhebungen und Farbschemaeinstellungen.

Bearbeiten → Alle auswählen (Strg+A)

Die gesamte Datei wird ausgewählt. Dies ist besonders zum Kopieren der gesamten Datei in eine andere Anwendung nützlich.

Bearbeiten → Auswahl aufheben (Strg+Umschalt+A)

Hebt eine vorhandene Auswahl auf.

Bearbeiten → Blockauswahlmodus (Strg+Umschalt+B)

Schaltet zwischen den beiden Arten des Auswahlmodus um. Wenn der Auswahlmodus **BLOCK** eingeschaltet ist, dann wird **[BLOCK]** in der Statusleiste angezeigt und Sie können Sie rechteckige Bereiche wie zum Beispiel die Spalten 5 bis 10 in den Zeilen 9 bis 15 auswählen.

Bearbeiten → Als HTML kopieren

Wechselt zwischen einem normalen und einem VI-ähnlichen modalen Bearbeitungsmodus Im Vi-Modus können viele Vi-Befehle für den Normalen und Visual-Modus benutzt werden. Außerdem kann für diesen Modus eine zusätzliche Statusleiste angezeigt werden, sie zeigt Befehle an, während sie eingegeben werden, sowie die Ausgabe von VI-Befehlen und den aktuellen Modus. Das Verhalten dieses Modus kann auf der Karteikarte VI-Eingabemodusder Seite **Bearbeitung** des Einrichtungsdialogs von KatePart eingestellt werden.

Bearbeiten → Überschreibmodus (Einfg)

Schaltet zwischen den beiden Arten des Eingabemodus um. Wenn der Modus **Einfügen** ist, dann setzen Sie die eingegebenen Zeichen an der Stelle des Cursors ein. Wenn der Modus **Überschreiben** ist, dann ersetzt jedes eingegebene Zeichen ein Zeichen rechts vom Cursor. Die Statusleiste zeigt den aktuellen Status des Auswahlmodus an, entweder **Einfügen** oder **Überschreiben**

Bearbeiten → Suchen ... (Strg+F)

Öffnet die Zusatzleiste für inkrementelle Suche im Editor-Fenster. An der linken Seite der Leiste finden Sie ein Symbol zum Schließen der Suchleiste, daneben befindet sich ein kleines Eingabefeld, in das Sie den gewünschten Suchbegriff eingeben können.

Sobald Sie Buchstaben in das Suchfeld eingeben, beginnt die Suche. Wenn eine entsprechende Textstelle gefunden wird, so wird die Fundstelle hervorgehoben und das Suchfeld wird hellgrün hinterlegt. Wenn der gesuchte Text nicht gefunden werden kann, wird das Suchfeld rot hinterlegt.

Mit den Knöpfen  und  können Sie zur nächsten bzw. vorherigen Fundstelle im Dokument springen.

Die im Dokument gefundenen Textstellen bleiben auch dann noch hervorgehoben, wenn Sie die Suchleiste schließen. Drücken Sie die Taste **Esc**, um die Hervorhebungen auszuschalten.

Wenn  aktiviert ist, wird die Groß- und Kleinschreibung beim Suchen beachtet.

Klicken Sie auf  auf der rechten Seite der Suchleiste, um zwischen der inkrementellen und der erweiterten Suche zu wechseln.

Bearbeiten → **Suchaktionen** → **Weitersuchen (F3)**

Wiederholt die zuletzt ausgeführte Suche, ohne dass die inkrementelle Suchleiste geöffnet wird und sucht dabei vorwärts von der aktuellen Cursorposition aus.

Bearbeiten → **Suchaktionen** → **Frühere suchen (Umschalt+F3)**

Wiederholt die zuletzt ausgeführte Suche, ohne dass die inkrementelle Suchleiste geöffnet wird und sucht dabei rückwärts.

Bearbeiten → **Suchaktionen** → **Auswahl suchen (Strg+H)**

Sucht das nächste Vorkommen des markierten Textes.

Bearbeiten → **Suchaktionen** → **Auswahl suchen (rückwärts) (Strg+Umschalt+H)**

Sucht nach dem vorherigen Vorkommen des markierten Textes.

Bearbeiten → **Ersetzen ... (Strg+R)**

Dieser Befehl öffnet den Dialog zum erweiterten Suchen und Ersetzen. Der Knopf oben links schließt die Leiste, daneben finden Sie das Eingabefeld für den Suchbegriff.

Sie können das Verhalten der Suche mit den Optionen **Einfacher Text**, **Ganze Wörter**, **Escape-Sequenzen** und **Regulärer Ausdruck** beeinflussen.

In den Modi **Escape-Sequenzen** und **Regulärer Ausdruck** wird der Knopf **Hinzufügen ...** unten im Kontextmenü verfügbar, über den Sie vordefinierte Zeichen/Ausdrücke für diese Modi aus einer Liste als Such- oder Ersetzungsmuster hinzufügen können.

Mit den Knöpfen  und  können Sie zur nächsten bzw. vorherigen Fundstelle im Dokument springen.

Geben Sie den Ersatztext in das Eingabefeld **Ersetzen** ein und drücken Sie anschließend den Knopf **Ersetzen** rechts daneben um die markierte Fundstelle entsprechend zu ersetzen, oder drücken Sie **Alle ersetzen**, um alle Fundstellen im ganzen Dokument zu ersetzen.

Sie können das Verhalten von Suchen und Ersetzen mit den verschiedenen Einstellungen unten in der Leiste ändern. Mit  werden nur Übereinstimmungen gefunden, die in Groß-/Kleinschreibung Ihrem Suchtext entsprechen. Mit  wird der Suchen/Ersetzen-Vorgang nur innerhalb des ausgewählten Textes ausgeführt. Die Einstellung **Alle hervorheben** bewirkt, dass alle Fundstellen farbig hinterlegt werden und die Anzahl der Fundstellen in einem kleinen Fenster angezeigt wird. (Auch, wenn noch keine Ersetzung stattgefunden hat. Dies kann nützlich sein, um auf einen Blick zu sehen, an welchen Stellen im Dokument Ersetzungen erfolgen würden, wenn Sie auf **Alle ersetzen** klicken.)

Klicken Sie auf  rechts in der Suchleiste, um von der erweiterten Suche zur einfachen Suche zu wechseln.

Bearbeiten → **Gehe zu** → **Zur passenden Klammer (Strg+6)**

Verschiebt den Cursor zur zugehörigen öffnenden oder schließenden Klammer.

Bearbeiten → **Gehe zu** → **Bis zur passenden Klammer markieren (Strg+Umschalt+6)**

Wählt den Text zwischen der öffnenden oder schließenden Klammer aus.

Bearbeiten → **Gehe zu** → **Zur vorherigen geänderten Zeile**

Zeilen, die seit dem Öffnen der Datei bearbeitet wurden, werden als geänderte Zeilen betrachtet. Diese Aktion geht zur vorherigen geänderten Zeile.

Bearbeiten → **Gehe zu** → **Zur nächsten geänderten Zeile**

Zeilen, die seit dem Öffnen der Datei bearbeitet wurden, werden als geänderte Zeilen betrachtet. Diese Aktion geht zur nächsten geänderten Zeile.

Bearbeiten → Gehe zu → Gehe zu Zeile ... (Strg+G)

Öffnet die Leiste „Gehe zu“, die zur Eingabe der Zeilennummer dient, an die der Cursor springen soll. Die Zeilennummer kann direkt eingegeben oder durch die Pfeile rechts im Eingabefeld erhöht oder verringert werden. Schließen Sie die Leiste wieder, indem Sie auf das Symbol auf der linken Seite klicken.

4.3 Das Menü Ansicht

Ansicht → Neues Fenster

Öffnet ein neues Fenster mit dem aktuellen Dokument. Alle Änderungen in einem der beiden Fenster werden auch in dem jeweils anderen Fenster erscheinen.

Ansicht → Auf Befehlszeile umschalten (F7)

Zeigt die Befehlszeile von KatePart am unteren Rand des Fensters an. Geben Sie hier **`help`** für die Hilfe und **`help list`** für eine Liste der verfügbaren Befehle ein. Weitere Informationen finden Sie im Kapitel Integrierte Befehlszeile im Editor.

Ansicht → Schrift vergrößern (Strg++)

Hiermit wird die Schriftgröße der Anzeige vergrößert.

Ansicht → Schrift verkleinern (Strg+-)

Hiermit wird die Schriftgröße der Anzeige verkleinert.

Ansicht → Schema

Dieses Menü enthält die verfügbaren Farbschemata. Sie können hier das Schema für die aktuelle Ansicht umschalten. Um das Standardschema zu verändern, benutzen Sie die Seite Schriften & Farben des Einrichtungsdialogs.

Ansicht → Zeilenumbruch → Dynamischer Zeilenumbruch (F10)

Dieser Befehl schaltet den dynamischen Zeilenumbruch ein und aus. Durch den dynamischen Zeilenumbruch wird der gesamte Text sichtbar, ohne dass horizontal gerollt werden muss, da der Inhalt einer Zeile wenn nötig in mehreren Zeilen angezeigt wird.

Ansicht → Zeilenumbruch → Anzeigen für dynamischen Zeilenumbruch

Wählen Sie hier, ob und wie die Markierungen für den dynamischen Zeilenumbruch angezeigt werden sollen. Dieser Menüpunkt steht nur zur Verfügung, wenn die Option **Zeilenumbruch** eingeschaltet ist.

Ansicht → Zeilenumbruch → Markierung für statischen Zeilenumbruch anzeigen

Wenn eingeschaltet, dann wird eine senkrechte Linie in der Spalte, an der der Zeilenumbruch erfolgt, angezeigt. Die Position wird in **Einstellungen → Editor einrichten ...** auf der Karte Bearbeitung festgelegt. Die Markierung wird nur dann angezeigt, wenn Sie eine Schrift mit fester Buchstabenbreite verwenden.

Ansicht → Randbereiche → Symbolspalte anzeigen (F6)

Mit diesem Eintrag wird ein zusätzlicher Rand an der linken Seite des aktiven Rahmens ein- oder ausgeschaltet, der Symbole anzeigen kann. Auf dem Symbolrand werden die Positionen von markierten Bereichen im Editor angezeigt.

Ansicht → Randbereiche → Zeilennummern anzeigen (F11)

Dieser Punkt schaltet die Anzeige einer Spalte mit den Zeilennummern am linken Rand des Editorfensters ein und aus.

Ansicht → Randbereiche → Markierung für Bildlaufleiste anzeigen

Wenn dieses Feld angekreuzt ist, dann werden im Dokument Markierungen in der senkrechten Bildlaufleiste angezeigt. Diese zeigen zum Beispiel Lesezeichen und entsprechen den Markierungen auf dem Symbolrand.

Ansicht → Randbereiche → Textgrafik auf Bildlaufleiste anzeigen

Dies ersetzt die Bildlaufleiste mit einer Grafik des Texts im aktuellen Dokument. Weitere Informationen über die Textgrafik auf der Bildlaufleiste finden Sie im Abschnitt Abschnitt 3.10.

Ansicht → Quelltextausblendung

Diese Einstellungen sind für die Quelltextausblendung vorhanden:

Markierungen für Quelltextausblendungen anzeigen (F9)

Schaltet die Anzeige der Quelltext-Ausblendungsleiste am linken Rand des Editorfensters ein und aus.

Aktuelle Ebene einklappen

Blendet die Ebene aus, die den Cursor enthält.

Aktuelle Ebene ausklappen

Blendet die Ebene ein, die den Cursor enthält.

Oberste Ebene einklappen (Strg+Umschalt+-)

Blendet alle Abschnitte der obersten Ebene im Dokument aus. Klicken Sie auf den nach rechts zeigenden Pfeil, um all Abschnitte in der obersten Ebene wieder einzublenden.

Oberste Ebene ausklappen (Strg+Umschalt++)

Blendet alle Abschnitte der obersten Ebene im Dokument ein.

Nicht druckbare Leerzeichen anzeigen

Umgebende Rahmen um nicht druckbare Leerzeichen anzeigen/ausblenden.

4.4 Das Menü Lesezeichen

Unterhalb der hier beschriebenen Einträge bekommt das Menü einen zusätzlichen Eintrag für jedes im aktuellen Dokument existierende Lesezeichen. Der Text des Lesezeicheneintrags besteht aus den ersten Wörter der durch das Lesezeichen markierten Zeile. Klicken Sie auf einen Lesezeicheneintrag, um den Cursor zu der markierten Zeile zu setzen. Der Editor rollt das Fenster, wenn notwendig.

Lesezeichen → Lesezeichen setzen (Strg+B)

Setzt oder entfernt ein Lesezeichen in der aktuellen Zeile des aktiven Dokuments. Wenn das Lesezeichen bereits existiert, wird es entfernt, wenn nicht, wird es gesetzt.

Lesezeichen → Alle Lesezeichen löschen

Dieser Befehl löscht alle Lesezeichen aus der Datei sowie die Lesezeichen-Liste am unteren Ende dieses Menüs.

Lesezeichen → Vorheriges (Alt+Bild auf)

Dieser Befehl bewegt den Cursor zur ersten Zeile mit Lesezeichen oberhalb der aktuellen Cursor-Postion. Der Menüeintrag enthält die Zeilennummer und den ersten Teil des Textes in der Zeile mit dem Lesezeichen. Der Menüeintrag ist nur verfügbar, wenn es oberhalb des Cursors eine Zeile mit Lesezeichen gibt.

Lesezeichen → Nächstes (Alt+Bild ab)

Dieser Befehl bewegt den Cursor zur ersten Zeile mit Lesezeichen unterhalb der aktuellen Cursor-Postion. Der Menüeintrag enthält die Zeilennummer und den ersten Teil des Textes in der Zeile mit dem Lesezeichen. Der Menüeintrag ist nur verfügbar, wenn es unterhalb des Cursors eine Zeile mit Lesezeichen gibt.

4.5 Das Menü Extras

Extras → **Nur-Lesen-Modus**

Setzt das aktuelle Dokument in dem Modus Nur-Lesen. Dies verhindert jegliche Änderungen am Dokument.

Extras → **Modus**

Wählen Sie hier den Dateityp, den Sie für das aktuelle Dokument verwenden wollen. Diese Einstellung überschreibt den unter **Einstellungen** → **Editor einrichten ...** auf der Karte Datentypen festgelegten Standardtyp für das aktuelle Dokument.

Extras → **Hervorhebung**

Wählen Sie hier das Hervorhebungsschema, das Sie für das aktuelle Dokument verwenden wollen. Diese Einstellung überschreibt die unter **Einstellungen** → **Editor einrichten ...** auf der Karte Hervorhebungen festgelegte Hervorhebungsregel für das aktuelle Dokument.

Extras → **Einrückung**

Wählen Sie hier den Einrückungsmodus, den Sie für das aktuelle Dokument verwenden wollen. Diese Einstellung überschreibt die unter **Einstellungen** → **Editor einrichten ...** auf der Karte Einrückung festgelegte Einrückungsregel für das aktuelle Dokument.

Extras → **Kodierung**

Wählen Sie hier die Kodierung, die Sie für das aktuelle Dokument verwenden wollen. Diese Einstellung überschreibt die unter **Einstellungen** → **Editor einrichten ...** auf der Seite **Öffnen/Speichern** festgelegte Standardkodierung nur für das aktuelle Dokument.

Extras → **Zeilenende**

Wählen Sie hier den Zeilenendemodus, den Sie für das aktuelle Dokument verwenden wollen. Diese Einstellung überschreibt den unter **Einstellungen** → **Editor einrichten ...** auf der Karte Öffnen/Speichern festgelegten Zeilenendemodus für das aktuelle Dokument.

Extras → **Byte-Reihenfolge-Markierung (BOM) hinzufügen**

Mit dieser Aktion kann ausdrücklich eine Byte-Reihenfolge-Markierung für Dokumente in Unicode-Kodierung hinzugefügt werden. Die Byte-Reihenfolge-Markierung ist ein Unicode-Zeichen, das die Bytereihenfolge (Big- oder Little-Endian) eines Textes anzeigt. Weitere Informationen finden Sie im Artikel Byte-Reihenfolge-Markierung.

Extras → **Skripte**

Dieses Untermenü enthält eine Liste von allen Aktionen, für die es Skripte gibt. Die Liste kann leicht durch das Schreiben eigener Skripte verändert werden. KatePart kann auf diesem Weg mit eigenen Werkzeugen erweitert werden.

Extras → **Skripte** → **Navigation**

Extras → **Skripte** → **Navigation** → **Cursor zur vorherigen passenden Einrückung verschieben (Alt+Umschalt+Pfeil hoch)**

Verschiebt den Cursor zur ersten Zeile oberhalb der aktuellen Cursor-Position, die die gleiche Einrückungstiefe wie die aktuelle Zeile hat.

Extras → **Skripte** → **Navigation** → **Cursor zur nächsten passenden Einrückung verschieben (Alt+Umschalt+Pfeil runter)**

Verschiebt den Cursor zur nächsten Zeile unterhalb der aktuellen Cursor-Position, die die gleiche Einrückungstiefe wie die aktuelle Zeile hat.

Extras → **Skripte** → **Bearbeitung**

Extras → **Skripte** → **Bearbeitung** → **Markierten Text sortieren**

Sortiert den ausgewählten Text oder das gesamte Dokument in aufsteigender Reihenfolge.

Extras → **Skripte** → **Bearbeitung** → **Zeilen nach unten verschieben (Strg+Umschalt+Pfeil runter)**
Verschiebt die markierten Zeilen nach unten.

Extras → **Skripte** → **Bearbeitung** → **Zeilen nach oben verschieben (Strg+Umschalt+Pfeil hoch)**
Verschiebt die markierten Zeilen nach oben.

Extras → **Skripte** → **Bearbeitung** → **Markierte Zeilen nach unten kopieren (Strg+Alt+Pfeil runter)**
Kopiert die markierten Zeilen nach unten.

Extras → **Skripte** → **Bearbeitung** → **Markierte Zeilen nach oben kopieren (Strg+Alt+Pfeil hoch)**
Kopiert die markierten Zeilen nach oben.

Extras → **Skripte** → **Bearbeitung** → **Markierten Text als URI kodieren**
Kodiert den ausgewählten Text, so dass dieser als Teil einer Abfrage in einer URL benutzt werden kann. Der Text ersetzt dabei die Auswahl in der URL.

Extras → **Skripte** → **Bearbeitung** → **Markierten Text als URI dekodieren**
Wenn ein Teil einer Abfrage in einer URL ausgewählt wird, dann wird diese Funktion die Auswahl durch den originalen Text ersetzen.

Extras → **Skripte** → **Emmet**

Extras → **Skripte** → **Emmet** → **Abkürzung ausschreiben**
Wandelt den ausgewählten Text in ein Paar von öffnenden und schließenden HTML- oder XML-Tags um. Ist zum Beispiel **`div`** ausgewählt, wird es durch `<div></div>` ersetzt.

Extras → **Skripte** → **Emmet** → **Umbruch mit Tag**
Bricht den ausgewählten Text mit dem auf der Befehlszeile angegebenen Tag um.

Extras → **Skripte** → **Emmet** → **Cursor zum passenden Tag verschieben**
Befindet sich der Cursor innerhalb eines öffnenden HTML- / XML-Tags, verschiebt diese Aktion ihn zum schließenden Tag. Befindet sich der Cursor innerhalb eines schließenden HTML- / XML-Tags, verschiebt diese Aktion ihn stattdessen zum öffnenden Tag.

Extras → **Skripte** → **Emmet** → **Inhalte von HTML/XML-Tag innerhalb wählen**
Befindet sich der Cursor innerhalb eines Paares von HTML- / XML-Tags, ändert diese Aktion die Auswahl zum Inhalt der HTML- / XML-Tags ohne die Tags selbst.

Extras → **Skripte** → **Emmet** → **Inhalte von HTML/XML-Tag außerhalb wählen**
Befindet sich der Cursor innerhalb eines Paares von HTML- / XML-Tags, ändert diese Aktion die Auswahl zum Inhalt der HTML- / XML-Tags einschließlich der Tags selbst.

Extras → **Skripte** → **Emmet** → **Kommentar ein-/ausschalten**
Ist die Auswahl kein Kommentar, wird diese Aktion die Auswahl in HTML- / XML-Kommentaren einschließen, z. B. `<!-- markierter Text -->`). Handelt es sich bei der Auswahl um einen Kommentar, werden die Tags stattdessen entfernt.

Extras → **Skripte** → **Emmet** → **Tag unter Cursor löschen**
Befindet sich der Cursor gerade innerhalb eines HTML- / XML-Tags, löscht diese Aktion das gesamte Tag.

Extras → **Skripte** → **Emmet** → **Zahl um 1 verringern**
Diese Aktion subtrahiert 1 vom aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **`5`** ausgewählt ist, wird dies zu `4` geändert.

Extras → **Skripte** → **Emmet** → **Zahl um 10 verringern**
Diese Aktion subtrahiert 10 vom aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **`15`** ausgewählt ist, wird dies zu `5` geändert.

Extras → **Skripte** → **Emmet** → **Zahl um 0.1 verringern**
Diese Aktion subtrahiert 0.1 vom aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **`4.5`** ausgewählt ist, wird dies zu `4.4` geändert.

Extras → **Skripte** → **Emmet** → **Zahl um 1 vergrößern**
Diese Aktion addiert 1 zum aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **5** ausgewählt ist, wird dies zu `6` geändert.

Extras → **Skripte** → **Emmet** → **Zahl um 10 vergrößern**
Diese Aktion addiert 10 zum aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **5** ausgewählt ist, wird dies zu `15` geändert.

Extras → **Skripte** → **Emmet** → **Zahl um 0.1 vergrößern**
Diese Aktion addiert 0.1 zum aktuell ausgewählten Text, sofern es sich dabei um eine Zahl handelt. Wenn zum Beispiel **4.5** ausgewählt ist, wird dies zu `4.6` geändert.

Extras → **Quelltextvervollständigung aufrufen (Strg+Leertaste)**

Manueller Aufruf der Quelltextvervollständigung, üblicherweise durch einen mit dieser Aktion belegten Kurzbefehl.

Extras → **Wortvervollständigung**

Mit **Wort unten erneut verwenden** (**Strg**+**9**) und **Wort oben erneut verwenden** (**Strg**+**8**) wird bei der Texteingabe vorwärts und rückwärts von der aktuellen Cursor-Position nach ähnlichen Wörtern gesucht und der Text vervollständigt. Mit **Shell-Eingabevervollständigung** wird ein Feld mit passenden Wörtern angezeigt.

Extras → **Rechtschreibung** → **Automatische Rechtschreibprüfung (Strg+Umschalt+O)**

Ist die **Automatische Rechtschreibprüfung** aktiviert, werden falsch geschriebene Wörter bei der Eingabe unterstrichen.

Extras → **Rechtschreibung** → **Rechtschreibung ...**

Ruft die Rechtschreibprüfung auf - ein Programm zum Finden und Korrigieren von Rechtschreibfehlern. Klicken auf diesen Menüeintrag startet das Programm zur Rechtschreibprüfung und öffnet dessen Dialogfenster, in dem Sie die Ausführung der Rechtschreibprüfung steuern können. Es gibt vier Textfelder in der Mitte des Dialogfensters mit den zugehörigen Namen links daneben. Diese sind von oben nach unten:

Unbekanntes Wort:
Hier zeigt das Programm zur Rechtschreibprüfung das aktuell als falsch erkannte Wort an. Dieses Wort hat das Programm zur Rechtschreibprüfung nicht in seinem Wörterbuch gefunden. Das Wörterbuch ist eine Datei, die eine Liste der korrekt geschriebenen Wörter enthält, mit denen das Programm zur Rechtschreibprüfung jedes einzelne Wort des zu prüfenden Textes vergleicht.

Ersetzen durch:
Wenn das Programm zur Rechtschreibprüfung ähnliche Wörter im Wörterbuch findet, wird das erste hier angezeigt. Der Benutzer kann den Vorschlag akzeptieren, eine eigene Korrektur eingeben oder einen anderen Vorschlag aus dem nächsten Feld auswählen.

Sprache:
Wenn Sie mehrere Wörterbücher installiert haben, können Sie hier das Wörterbuch oder die Sprache auswählen.

Auf der rechten Seite des Dialogfensters befinden sich sechs Knöpfe, mit denen Sie den Prüfungsvorgang steuern können. Diese sind im einzelnen:

Zum Wörterbuch hinzufügen
Drücken dieses Knopfes fügt das Wort im Feld **Unbekanntes Wort:** zum benutzten Wörterbuch des Rechtschreibprüfungsprogramms hinzu. Das bedeutet, dass dieses Wort in Zukunft immer als richtig geschrieben erkannt wird.

Vorschläge

Das Prüfungsprogramm zeigt hier eine Liste mit möglichen Vorschlägen an, die das gefundene Wort ersetzen können. Klicken auf einen dieser Vorschläge trägt diesen Vorschlag in das Feld **Ersetzen durch:** gleich darüber ein.

Ersetzen

Ersetzt das gefundene Wort im Dokument mit dem Wort im Feld **Ersetzen durch:**.

Alle ersetzen

Dieser Knopf ersetzt nicht nur das aktuell **Unbekanntes Wort** mit dem Inhalt des Feldes **Ersetzen durch:**, sondern ersetzt automatisch alle Vorkommen des unbekannten Wortes im Dokument.

Ignorieren

Die Rechtschreibprüfung wird ohne Änderungen am aktuellen Wort fortgesetzt.

Alle ignorieren

Klicken auf diesen Knopf setzt die Rechtschreibprüfung ohne Änderungen **unbekannten Wortes** fort und ignoriert alle weiteren Vorkommen dieses Wortes im gesamten Dokument.

> ANMERKUNG
> Dies gilt nur für den aktuellen Lauf der Rechtschreibprüfung, wenn später noch einmal die Rechtschreibung geprüft wird, wird dasselbe Wort wieder als falsch erkannt.

Drei weitere Knöpfe befinden sich im unteren Bereich des Dialogs:

Hilfe

Dies startet das KDE-Hilfesystem mit den Hilfeseiten zu diesem Dialog.

Abgeschlossen

Dieser Knopf beendet die Rechtschreibprüfung und kehrt zum Dokument zurück.

Abbrechen

Dieser Knopf bricht die Rechtschreibprüfung ab. Alle Änderungen werden vor der Rückkehr zum Dokument zurückgenommen.

Extras → Rechtschreibung → Rechtschreibung (ab Cursor) ...

Ruft das Rechtschreibprüfungsprogramm auf - mit dem Unterschied, dass die Prüfung an der aktuellen Cursor-Position beginnt und nicht am Anfang des Dokuments.

Extras → Rechtschreibung → Rechtschreibprüfung für Auswahl ...

Ruft das Rechtschreibprüfungsprogramm auf und prüft den aktuell ausgewählten Text.

Extras → Rechtschreibung → Wörterbuch auswählen ...

Zeigt ein Auswahlfeld mit allen verfügbaren Wörterbüchern zur Rechtschreibprüfung unten im Editorfenster an. Dadurch kann das Wörterbuch schnell gewechselt werden, z. B. für die automatische Rechtschreibprüfung von Text in mehreren Sprachen.

Extras → Einrückungen löschen

Löscht die Einrückung für die aktuelle Auswahl oder für die Zeile, in der sich der Cursor befindet. Löschen der Einrückungen stellt sicher, dass der gesamte ausgewählte Text nach dem ausgewählten Einrückungsmodus behandelt wird.

Extras → Ausrichten

Bewirkt, dass die aktuelle Zeile oder aktuelle Auswahl nach den Einstellungen für den aktuellen Einrückungsmodus und den Einrückungseinstellungen im Dokument neu ausgerichtet wird.

Extras → Kommentar (Strg+D)

Dieser Befehl fügt ein Leerzeichen am Zeilenanfang der aktuellen Zeile oder an den Zeilenanfängen aller markierten Zeilen ein.

Extras → **Kommentar entfernen (Strg+Umschalt+D)**

Dieser Befehl entfernt (wenn vorhanden) ein Leerzeichen vom Zeilenanfang der aktuellen Zeile oder von den Zeilenanfängen der markierten Zeilen.

Extras → **Großschreibung (Strg+U)**

Setzt den ausgewählten Text oder den Buchstaben nach dem Cursor in Großbuchstaben.

Extras → **Kleinschreibung (Strg+Umschalt+U)**

Setzt den ausgewählten Text oder den Buchstaben nach dem Cursor in Kleinbuchstaben.

Extras → **Großschreibung am Wortanfang (Strg+Alt+U)**

Setzt den ausgewählten Text oder das aktuelle Wort in Großbuchstaben.

Extras → **Zeilen zusammenführen (Strg+J)**

Setzt die ausgewählten Zeilen oder die aktuelle und die nächste Zeile zusammen. Ein Leerzeichen wird zwischen die Zeileninhalte gesetzt. Noch vorhandene weitere Leerzeichen werden an den betroffenen Zeilenanfängen oder -enden entfernt.

Extras → **Zeilenumbruch hinzufügen**

Das gesamte Dokument wird automatisch mit Zeilenumbrüchen versehen. Das heißt, dass automatisch eine neue Zeile begonnen wird, wenn die aktuelle Zeile die Länge, die unter **Zeilenumbruch bei** auf der Karte Bearbeitung in **Einstellungen** → **Editor einrichten ...** eingestellt wurde, überschritten wird.

4.6 Die Menüs „Einstellungen” und „Hilfe”

KatePart benutzt die bekannten KDE-Menüeinträge **Einstellungen** und **Hilfe**. Mehr dazu erfahren Sie in den Abschnitten zu den Menüs Einstellungen und Hilfe in den KDE-Grundlagen.

Kapitel 5

Weiterentwickelte Editierwerkzeuge

Anders Lund
Dominik Haumann
GUI-Übersetzung: Thomas Diehl
Deutsche Übersetzung: Matthias Schulz

5.1 Kommentar/Kommentar entfernen

Die Befehle **Kommentar** und **Kommentar entfernen** im Menü **Bearbeiten** erlauben das Hinzufügen oder Entfernen von Kommentarzeichen zur Auswahl, oder der aktuellen Textzeile, wenn kein Text markiert wurde. Diese Funktionen stehen nur zur Verfügung, wenn das benutzte Textformat Kommentare unterstützt.

Die Regeln für Kommentare werden in den Definitionen für die Syntax festgelegt, wenn Hervorhebungen für Syntax nicht benutzt werden, ist die Nutzung der Befehle also nicht möglich.

Einige Formate nutzen Kommentarzeichen für einzelne Zeilen, manche nutzen Kommentarzeichen für mehrere Zeilen, manche beides. Wenn Kommentarzeichen für mehrere Zeilen nicht verfügbar sind, kann eine Auswahl nicht auskommentiert werden, deren letzte Zeile nicht vollständig in die Auswahl einbezogen ist.

Wenn Kommentarzeichen für einzelne Zeilen definiert sind, werden diese bevorzugt eingesetzt, dies hilft, Probleme mit eingebetteten Kommentaren zu vermeiden.

Wenn Sie Kommentarzeichen entfernen, sollte nur kommentierter Text ausgewählt sein. Wenn mehrzeilige Kommentare entfernt werden, werden Leerzeichen und Tabulatoren (whitespaces) außerhalb der Kommentarzeichen ignoriert.

Um Kommentarzeichen einzufügen, wählen Sie im Menü **Extras** → **Kommentar** oder das Tastenkürzel, hier standardmäßig **Strg**+**D**.

Zum Entfernen von Kommentarzeichen wählen Sie im Menü **Bearbeiten** → **Kommentar entfernen** oder das Tastenkürzel, hier standardmäßig **Strg**+**Umschalt**+**D**.

5.2 Die integrierte Befehlszeile im Editor

KatePart;s Editorkomponente hat eine interne Befehlszeile, von der aus Sie verschiedene Aktionen von einer minimalen GUI ausführen können. Die Befehlszeile ist ein Texteingabefeld am unteren Rand des Editorbereichs. Sie können diese einblenden, indem Sie im Menü **Ansicht** → **Auf Befehlszeile umschalten** wählen oder das Tastenkürzel verwenden (standardmäßig ist **F7** eingestellt). Der Editor stellt einige Befehle bereit, die nachfolgend beschrieben werden. Außerdem können Module weitere Befehle bereitstellen.

Um einen Befehl auszuführen, geben Sie diesen in die Befehlszeile ein und drücken Sie Eingabetaste. Der Befehl gibt aus, ob die Ausführung erfolgreich war, eventuell wird noch eine Mitteilung ausgegeben. Wenn Sie den Befehl durch Drücken des Tastenkürzels **F7** eingegeben haben, wird die Befehlszeile nach einigen Sekunden ausgeblendet. Um die Mitteilung zu löschen und einen neuen Befehl einzugeben, drücken Sie das Tastenkürzel **F7** noch einmal.

Die Befehlszeile hat ein eingebautes Hilfesystem, das durch den Befehl **help** aufgerufen wird. Der Befehl **help list** zeigt eine Liste aller verfügbaren Befehle an, Hilfe zu einem speziellen Befehl erhalten Sie durch Eingabe von **help** ***befehl***.

Die Befehlszeile hat eine eingebauten Verlaufsspeicher, sodass Sie bereits eingegebene Befehle wiederverwenden können. Um aus den bisherigen Befehlen auszuwählen, benutzen Sie die Tasten **Pfeil hoch** und **Pfeil runter**. Wenn bisherige Befehle angezeigt werden, dann ist automatisch der Teil des Befehls, der die Argumente enthält, markiert, sodass Sie die Argumente sofort überschreiben können.

5.2.1 Standardbefehle der Befehlszeile

TYPEN DER ARGUMENTE

BOOLEAN
: Dieser Typ wird mit Befehlen benutzt, die Dinge ein- und ausschalten. Zulässige Werte sind **`on`**, **`off`**, **`true`**, **`false`**, **`1`** oder **`0`**.

INTEGER
: Eine ganze Zahl.

STRING
: Ein Text, umschlossen von einfachen Anführungszeichen (') oder von doppelten Anführungszeichen ("), wenn der Text Leerzeichen enthält.

5.2.1.1 Befehle zum Einrichten des Editors

Diese Befehle werden von der Editorkomponente bereitgestellt und gestatten das Einrichten des Editors für die aktuelle Ansicht des aktuellen Dokuments. Dies ist hilfreich, wenn Sie von den Standardeinstellungen abweichende Einstellungen, z. B. für Einrückungen benutzen wollen.

`set-tab-width` **INTEGER Weite**
: Setzt die Tabulatorweite auf **`Weite`**.

`set-indent-width` **INTEGER Weite**
: Setzt die Einrückungsweite auf **`Weite`**. Dieser Wert wird nur benutzt, wenn Sie Leerzeichen zum Einrücken verwenden.

`set-word-wrap-column` **INTEGER Weite**
: Setzt die Zeilenlänge für den harten Zeilenumbruch auf **`Weite`**. Dieser Wert wird nur benutzt, wenn Sie den automatischen Zeilenumbruch benutzen.

`set-icon-border` **BOOLEAN enable**
: Schaltet die Anzeige des Symbolrandes ein und aus.

`set-folding-markers` **BOOLEAN enable**
: Schaltet die Markierungen für die Quelltextausblendung ein und aus.

`set-line-numbers` **BOOLEAN enable**
: Schaltet die Zeilennummerierung ein und aus.

`set-replace-tabs` BOOLEAN enable

Wenn eingeschaltet, werden Tabulatorzeichen durch Leerzeichen ersetzt.

`set-remove-trailing-space` BOOLEAN enable

Wenn eingeschaltet, werden Leerzeichen und andere Zwischenräume am Zeilenanfang entfernt, wenn der Cursor eine Zeile verlässt.

`set-show-tabs` BOOLEAN enable

Wenn eingeschaltet, werden Tabulatorzeichen und vorangestellte Leerzeichen durch kleine Punkte dargestellt.

`set-show-indent` BOOLEAN enable

Wenn eingeschaltet, wird die Einrückung durch eine punktierte Linie dargestellt.

`set-indent-spaces` BOOLEAN enable

Wenn eingeschaltet, werden Leerzeichen in der mit `indent-width` eingestellten Anzahl für jedes Einrückungsniveau benutzt und nicht Tabulatorzeichen.

`set-mixed-indent` BOOLEAN enable

KatePart benutzt eine Mischung aus Tabulatoren und Leerzeichen zum Einrücken wenn diese Option eingeschaltet ist. Jedes Einrückungsniveau ist `indent-width` breit, es werden möglichst viele Leerzeichen durch Tabulatoren ersetzt.

Wenn dieser Befehl ausgeführt wird, wird außerdem das Einrücken mit Leerzeichen eingeschaltet und wenn die Einrückungsbreite noch nicht festgelegt ist, dann wird diese auf die Hälfte von `tab-width` für dieses Dokument zum Ausführungszeitpunkt gesetzt.

`set-word-wrap` BOOLEAN enable

Schaltet dynamischen Zeilenumbruch ein oder aus.

`set-replace-tabs-save` BOOLEAN enable

Wenn eingeschaltet, werden Tabulatorzeichen durch Leerzeichen ersetzt, wenn das Dokument gespeichert wird.

`set-remove-trailing-space-save` BOOLEAN enable

Wenn eingeschaltet, werden am Anfang der Zeile stehende Leerzeichen oder Tabulatorzeichen entfernt, wenn das Dokument gespeichert wird.

`set-indent-mode` STRING name

Setzt den Einrückungsmodus auf **`name`**. Wenn **`name`** nicht angegeben ist oder ein ungültiger Name verwendet wurde, wird der Modus ‚Kein(none)‘ gesetzt. Verfügbare Modi sind: ’none’, ’normal’, ’cstyle’, ’haskell’, ’lilypond’, ’lisp’, ’python’, ’ruby’ und ’xml’.

`set-auto-ident` BOOLEAN script

Aktiviert oder deaktiviert die automatische Einrückung.

`set-highlight` STRING highlight

Setzt den Hervorhebungsmodus für das aktuelle Dokument. Das Argument muss ein gültiger Name für einen Hervorhebungsmodus sein. Die gültigen Modi findet man unter **Extras** → **Hervorhebungen**. Dieser Befehl zeigt eine Liste der möglichen Argumente an, wenn die ersten Zeichen des Argumentes eingegeben wurden.

`reload-scripts`

Lädt alle von Kate benutzten JavaScript-Skripte neu, einschließlich der Skripte für die Befehlszeile und Einrückung.

`set-mode` STRING mode

Setzt das Dateityp-Schema für das aktuelle Dokument.

`nn[oremap]` STRING original STRING mapped

Weist den Kurzbefehl **`original`** dem Kurzbefehl **`mapped`** zu.

5.2.1.2 Befehle zum Bearbeiten

Diese Befehle bearbeiten und verändern das aktuelle Dokument.

indent
Rückt markierten Text oder die aktuelle Zeile ein.

unindent
Hebt die Einrückung für die markierten Zeilen oder die aktuelle Zeile auf.

cleanindent
Setzt die Einrückungen in den markierten Zeilen oder in der aktuellen Zeile in den Grundzustand zurück. Hierzu werden die Einstellungen für das aktuelle Dokument verwendet.

comment
Setzt Kommentarzeichen um die markierten Zeilen oder die aktuelle Zeile zu Kommentaren zu machen. Es wird das Kommentarzeichen aus der Hervorhebungsdefinition für das aktuelle Textformat benutzt.

uncomment
Entfernt Kommentarzeichen von den markierten Zeilen oder der aktuellen Zeile. Es wird das Kommentarzeichen aus der Hervorhebungsdefinition für das aktuelle Textformat benutzt.

kill-line
Löscht die aktuelle Zeile

replace STRING suchtext STRING replacement

Ersetzt den mit **suchtext** übereinstimmenden Text durch **replacement**. Wenn Sie Leerzeichen oder Tabulatoren im **suchtext** verwenden wollen, dann müssen Sie sowohl **such text** als auch **replacement** in einfache oder doppelte Anführungszeichen einschließen. Wenn Worte nicht in Anführungszeichen eingeschlossen sind, wird das erste Wort als **suc htext** und der Rest als **replacement** benutzt. Wenn **replacement** nicht angegeben ist, dann wird jedes Auftreten von **suchtext** gelöscht.

Sie können bestimmte Einstellungen für die Suche vornehmen, indem Sie einen Doppelpunkt, gefolgt von einem oder mehreren Buchstaben - die die Einstellungen enthalten - anfügen. Die Form der Eingabe ist dann: **replace:options suchtext replacement**. Folgende Einstellungen sind verfügbar:

b
Rückwärts suchen.

c
Suchen ab Cursor-Position.

e
Suchen nur in markiertem Text.

r
Suche nach einem regulären Ausdruck. Wenn diese Einstellung verwendet wird, können Sie **\N** im Ersetzungstext verwenden, die Anzahl, wie oft der Suchtext gefunden wurde, wird dann in den Ersetzungstext eingefügt.

s
Suche unter Berücksichtigung von Groß- und Kleinschreibung.

p
Nachfragen vor Ersetzen des nächsten Auftretens.

w
Nur ganze Wörter erfüllen die Suchbedingung.

date STRING format
Setzt das aktuelle Datum und die aktuelle Uhrzeit im durch **format** eingestellten Format ein. Wenn kein Format eingestellt wurde, wird das Format „yyyy-MM-dd hh:mm:ss" als Standard benutzt. Die folgenden Übersetzungen werden vorgenommen, wenn **format** ausgewertet wird:

d	Tag als Ziffer ohne führende Null (1-31).
dd	Tag als Ziffer mit führender Null (01-31).
ddd	Tag als abgekürzter Name in Landessprache (z. B. ‚Mon‘..‚Son‘).
dddd	Tag als langer Name in Landessprache (z. B. ‚Montag‘..‚Sonntag‘).
M	Monat als Ziffer ohne führende Null (1-12).
MM	Monat als Ziffer mit führender Null (01-12).
MMMM	Der lange, lokale Monatsname (z. B. „Januar“..„Dezember“).
MMM	Monat als abgekürzter Name in Landessprache (z. B.‚Jan‘..‚Dez‘).
yy	Das Jahr als zweistellige Ziffer (00-99).
yyyy	Das Jahr als vierstellige Ziffer (1752-8000).
h	Stunde ohne führende Null (0..23 oder 1..12 wenn AM/PM verwendet wird).
hh	Stunde mit führender Null (00..23 oder 00..12 wenn AM/PM verwendet wird).
m	Minute ohne führende Null (0.59).
mm	Minute mit führender Null (00..59).
s	Sekunde ohne führende Null (0.59).
ss	Sekunde mit führender Null (00.59).
z	Millisekunden ohne führende Null (0..999).
zzz	Millisekunden mit führender Null (000..999).
AP	Benutzt die Anzeige mit AM/PM. AP wird entweder von „AM” oder „PM” ersetzt.
ap	Benutzt die Anzeige mit am/pm. ap wird durch „am” oder „pm” ersetzt.

`char` STRING identifier

Dieser Befehl erlaubt das Einsetzen von Zeichen in Text durch die Eingabe Ihrer Kodierung in dezimaler, oktaler oder hexadezimaler Form. Rufen Sie die Befehlszeile auf, schreiben Sie in das Eingabefeld das Wort **`char:[nummer]`** und klicken Sie auf **OK**.

Beispiel 5.1 Beispiele zu char

Eingabe: **`char:234`**
Ausgabe: ê
Eingabe: **`char:0x1234`**
Ausgabe: jué

s///[ig] %s///[ig]

Dieser Befehl führt Suchen/Ersetzen auf der aktuellen Zeile oder in der gesamten Datei aus (**%s///**).

Kurz gesagt, der Text wird nach dem *Suchtext*, dem regulären Ausdruck zwischen dem ersten und dem zweiten Schrägstrich, durchsucht und wenn der *Suchtext* gefunden wurde, wird der übereinstimmende Teil des Textes durch den Ausdruck zwischen dem zweiten und dem hinterem Schrägstrich ersetzt. Runde Klammern im *Suchtext* erzeugen *Referenzen*, die dann später dazu benutzt werden, die Zeichenfolgen wiederzuverwenden. Diese Referenzen werden wie folgt aufgerufen: **\1** für die erste Referenz, **\2** für die zweite und so weiter.

Um nach einem Sonderzeichen, `(` oder `)`, zu suchen, müssen Sie dieses durch einen Rückwärtsschrägstrich kenntlich machen. **\(\)**

Wenn Sie ein **i** an das Ende des Ausdruckes anhängen, wird beim Suchen nicht nach Groß- und Kleinbuchstaben unterschieden. Das Anhängen eines **g** legt fest, dass alle Vorkommen des Suchtextes ersetzt werden, normalerweise wird nur das erste Vorkommen ersetzt.

Beispiel 5.2 Ersetzen von Text in der aktuellen Zeile

Ihr lieber Computer verweigerte gerade die Ausführung eines Programms, mit der Bemerkung, dass die Klasse `myClass`, die in der Zeile 3902 im Quelltext verwendet wird, nicht definiert ist. ″Natürlich!″, denken Sie, das muss `MyClass` heißen. Sie gehen zur Zeile 3092, rufen den Bearbeitungsbefehl auf, geben **s/myclass/MyClass/i** ein und klicken auf **OK**, Speichern die Datei und kompilieren – ohne Fehlermeldungen.

Beispiel 5.3 Ersetzen von Text in der gesamten Datei

Stellen Sie sich vor, Sie hätten eine Datei, in der eine „Miss Jensen” einige Male erwähnt wird. Jemand kommt zur Tür herein und erzählt Ihnen, dass sie gerade „Mr Jones” geheiratet hat. Sie stehen nun vor der Aufgabe, jedes „Miss Jensen” durch „Ms Jones” zu ersetzen.
Rufen Sie die Befehlszeile auf, geben Sie **%s/Miss Jensen/Ms Jones/** ein drücken Sie die Eingabetaste - fertig.

Beispiel 5.4 Ein etwas komplizierteres Beispiel

Dieses Beispiel benutzt *Referenzen* und auch eine *Wortklasse* (wenn Sie nicht wissen, was das bedeutet, sehen Sie bitte in der unten angegebenen Dokumentation nach).
Stellen Sie sich vor, Sie hätten folgende Zeile:

```
void MyClass::DoStringOps( String      &foo, String &bar, String *p, int  & ↩
    a, int &b )
```

Sie erkennen, dass dies nicht gut lesbar ist und entscheiden, das Schlüsselwort `const` für alle „address of"-Argumente zu verwenden (diese sind durch den vorangestellten Operator & gekennzeichnet). Außerdem wollen Sie die Zwischenräume vereinfachen, sodass nur noch ein Leerzeichen zwischen den Wörtern steht.
Rufen Sie den Bearbeitungsbefehl auf, geben Sie **`s/\s+(\w+)\s+(&)/ const \1 \2/g`** und klicken Sie auf **OK**. Das **`g`** am Ende des regulären Ausdrucks bewirkt, dass der reguläre Ausdruck jedesmal neu kompiliert wird, um die *Referenz* zu sichern.
Ausgabe: `void MyClass::DoStringOps( const String &foo, const String &bar, String *p , const int &a, const int &b )`
Erledigt! Was passierte hier? Es wurde nach Leerzeichen (`\s+`) gefolgt von einem oder mehreren alphanumerischen Zeichen (`\w+`) gefolgt von einem oder mehreren Leerzeichen (`\s+`) gefolgt von einem Ampersand (&) gesucht und dabei der alphanumerische Abschnitt und das Ampersand (&) gesichert, um diese beim Ersetzen wiederzuverwenden. Dann haben wir den übereinstimmenden Teil der Zeile ersetzt durch ein Leerzeichen gefolgt von „const" gefolgt von einem Leerzeichen gefolgt vom gesicherten Abschnitt (`\1`) gefolgt von einem Leerzeichen gefolgt vom gesicherten Ampersand (&) (`\2`).
In einigen Fällen war der gesicherte Abschnitt „String", in einigen „int", sodass das Benutzen der Wortklasse `\w` und des `+`-Zeichens zum Angeben von „ein oder mehrere" sich als wertvoll erwies.

`sort`
: Sortiert den markierten Text oder das ganze Dokument.

`natsort`
: Sortiert den markierten Text oder das ganze Dokument in natürlicher Reihenfolge.

Beispiel 5.5 sort vs. natsort

`sort(a10, a1, a2)` ergibt `a1, a10, a2`
`natsort(a10, a1, a2)` ergibt `a1, a2, a10`

`moveLinesDown`
: Verschiebt die markierten Zeilen nach unten.

`moveLinesUp`
: Verschiebt die markierten Zeilen nach oben.

`uniq`
: Entfernt doppelte Zeilen aus dem markierten Text oder dem gesamten Dokument.

`rtrim`
: Entfernt Leerzeichen am Zeilenende aus dem markierten Text oder dem gesamten Dokument.

`ltrim`
: Entfernt Leerzeichen am Zeilenanfang aus dem markierten Text oder dem gesamten Dokument.

`join` [STRING separator]

Verbindet die ausgewählten Zeilen oder das gesamte Dokument. Wahlweise kann ein Parameter mit der Definition des Trennzeichens angegeben werden, zum Beispiel **`join ', '`**

`rmblank`

Entfernt alle Leerzeichen aus dem markierten Text oder dem gesamten Dokument.

`alignon`

Dieser Befehl richtet Zeilen in der Textauswahl oder dem gesamten Dokument an einer Spalte aus, welche durch einen regulären Ausdruck als Argument angegeben wird.

Wenn Sie ein leeres Muster angeben, wird standardmäßig am ersten nicht-leeren Zeichen ausgerichtet.

Wenn das Muster einen Treffer hat, wird an dessen Übereinstimmung ausgerichtet.

Beispiele:

`alignon -` fügt Leerzeichen vor dem ersten „-“ in jeder Zeile ein, um diese an derselben Spalte auszurichten.

`alignon :\\s+(.)` fügt Leerzeichen vor dem ersten nicht leeren Zeichen ein, das auf einen Doppelpunkt folgt, um die Zeilen an dessen Spalte auszurichten.

`unwrap`

Entfernt den Zeilenumbruch aus dem markierten Text oder dem gesamten Dokument.

`each` STRING script

Mit einer JavaScript-Funktion als Argument wird diese Funktion für die Liste der ausgewählten Zeilen aufgerufen und die Zeilen mit dem Rückgabewert der Funktion ersetzt

Beispiel 5.6 Verbindet die ausgewählten Zeilen

```
each 'function(lines){return lines.join(˝, ˝)}'
```

Oder noch kürzer:

```
each 'lines.join(˝, ˝)'
```

`filter` STRING script

Mit einer JavaScript-Funktion als Argument wird diese Funktion für die ausgewählten Zeilen aufgerufen und die Zeilen werden entfernt, bei denen die Funktion den Wert Falsch zurück gibt

Beispiel 5.7 Entfernt leere Zeilen.

```
filter 'function(1){return 1.length > 0;}'
```

Oder noch kürzer:

```
filter 'line.length > 0'
```

`map` STRING script

Mit einer JavaScript-Funktion als Argument wird diese Funktion für die Liste der ausgewählten Zeilen aufgerufen und die Zeile mit dem Rückgabewert der Funktion ersetzt

Beispiel 5.8 Entfernt leere Zeilen.

```
map 'function(line){return line.replace(/^s+/,˝˝);}'
```

Oder noch kürzer:

```
map 'line.replace(/^s+/,˝˝)'
```

`duplicateLinesUp`

Verdoppelt die ausgewählten Zeilen oberhalb der aktuellen Auswahl.

`duplicateLinesDown`
Verdoppelt die ausgewählten Zeilen unterhalb der aktuellen Auswahl.

5.2.1.3 Befehle zur Bewegung im Dokument

`goto` INT line
Dieser Befehl setzt den Cursor auf die angegebene Zeile.

`grep` STRING pattern
Sucht im Dokument nach dem regulären Ausdruck **`muster`**. Weitere Informationen finden Sie unter Anhang A.

`find` STRING pattern
Dieser Befehl setzt den Cursor auf das erste Auftreten des **`Suchtext`** entsprechend der Einstellungen. Weitere Fundstellen werden durch **Bearbeiten** → **Weitersuchen** oder Drücken des Tastenkürzels (Standard ist **F3**) gefunden.

Der Befehl **`find`** kann durch das Anhängen eines Doppelpunktes und eines oder mehrerer Buchstaben in der Form **`find:options pattern`** ergänzt werden. Die folgenden Einstellungen sind verfügbar:

`b`
Rückwärts suchen.
`c`
Suchen ab Cursor-Position.
`e`
Suchen nur in markiertem Text.
`r`
Suche nach einem regulären Ausdruck. Wenn diese Einstellung verwendet wird, können Sie **`\N`** im Ersetzungstext verwenden, die Anzahl, wie oft der Suchtext gefunden wurde, wird dann in den Ersetzungstext eingefügt.
`s`
Suche unter Berücksichtigung von Groß- und Kleinschreibung.
`w`
Nur ganze Wörter erfüllen die Suchbedingung.

`ifind` STRING pattern
Dieser Befehl sucht „schon beim Eingeben" nach dem Suchtext. Sie können auch hier die Suche durch Anhängen eines Doppelpunktes und eines oder mehrerer Buchstaben in ihrem Verhalten anpassen. Die Eingabe muss dann in der Form: **`ifind:options suchtext`** erfolgen. Die folgenden Einstellungen stehen zur Verfügung:

`b`
Rückwärts suchen.
`r`
Suche nach einem regulären Ausdruck.
`s`
Suche unter Berücksichtigung von Groß- und Kleinschreibung.
`c`
Suchen ab Cursor-Position.

5.2.1.4 Befehle für die grundlegenden Editor-Funktionen. Diese hängen von der Anwendung ab, in der die Editorkomponente verwendet wird.

w
: Speichert das aktuelle Dokument.

wa
: Speichert alle gerade geöffneten Dateien.

q
: Schließt das aktuelle Dokument.

qa
: Schließt alle geöffneten Dokumente.

wq
: Speichert und schließt das aktuelle Dokument.

wqa
: Speichert und schließt alle geöffneten Dokumente.

x
: Speichert und schließt das aktuelle Dokument nur, wenn es geändert wurde.

x
: Speichert und schließt alle geöffneten Dokument nur, wenn sie geändert wurden.

bp
: Geht zum vorherigen Dokument in der Dokumentliste.

bn
: Geht zum nächsten Dokument in der Dokumentliste.

new
: Öffnet ein neues Dokument in waagerecht geteilter Ansicht.

vnew
: Öffnet ein neues Dokument in senkrecht geteilter Ansicht.

e
: Lädt das aktuelle Dokument erneut, wenn es auf dem Datenträger geändert wurde.

enew
: Bearbeitet ein neues Dokument.

print
: Öffnet den Druckdialog, um das aktuelle Dokument zu drucken.

5.3 Benutzen von Quelltextausblendung

Quelltextausblendung dient zum Verstecken von Teilen des Dokuments im Editor, sodass große Dokumente einfacher zu lesen sind. In KatePart werden die ausblendbaren Abschnitte unter Zugrundelegung der Hervorhebungsregeln ermittelt und demzufolge sind Quelltextausblendungen nur in manchen Formaten verfügbar. Dies sind besonders Quelltexte in Programmiersprachen, XML und Ähnliches. Die meisten Hervorhebungsregeln, die Quelltextausblendungen bereitstellen, lassen auch die manuelle Festlegung von ausblendbaren Abschnitten zu, üblicherweise werden dazu die Schlüsselwörter **BEGIN** und **END** benutzt.

Um die Funktion Quelltextausblendung zu benutzen, wählen Sie im Menü **Ansicht** → **Markierungen für Quelltextausblendungen anzeigen**. Es wird dann am linken Rand des Editorfensters ein grauer Rand eingeblendet, der eine grafische Darstellung der ausblendbaren Abschnitte enthält. In diesen Markierungen sind Symbole enthalten, die die möglichen Operationen anzeigen. Wenn zum Beispiel ein Dreieck mit der Spitze nach unten angezeigt wird, kann dieser Abschnitt ausgeblendet werden, ein nach rechts zeigendes Dreieck dagegen zeigt, dass hier ein Abschnitt ausgeblendet wurde. Dieser kann durch Klicken auf das Dreiecksymbol wieder eingeblendet werden.

Drei Befehle sind im Menü enthalten, die die Quelltextausblendung beeinflussen, sehen Sie in der Menü-Dokumentation für weitere Einzelheiten nach.

Der Status der Quelltextausblendung wird gespeichert, wenn eine Datei geschlossen wird. Öffnen Sie diese Datei erneut, wird sie mit den ausgeblendeten Ebenen wie beim Schließen angezeigt. Auch beim erneuten Laden einer Datei bleibt die eingestellte Quelltextausblendung erhalten.

Wenn Sie keine Quelltextausblendung benutzen wollen, dann können Sie die Funktion **Markierung für Quelltextausblendungen anzeigen** auf der Seite Erscheinungsbild in den Einstellungen komplett ausschalten.

Kapitel 6

KatePart erweitern

T.C. Hollingsworth
GUI-Übersetzung: Thomas Diehl
Deutsche Übersetzung: Matthias Schulz

6.1 Einführung

Wie jeder gute Texteditor bietet auch KatePart verschiedene Möglichkeiten für Erweiterungen. Sie können Skripte in JavaScript schreiben, um Funktionen zu erweitern. Wenn Sie dann KatePart erweitert haben, laden wir Sie ein, Ihre Verbesserungen mit der ganzen Welt zu teilen.

6.2 Arbeiten mit Syntaxhervorhebungen

6.2.1 Überblick

Syntaxhervorhebungen bewirken, dass der Editor den Text automatisch in verschiedenen Farben und Schriftstilen anzeigt, abhängig von der Funktion der Zeichenfolge in Beziehung zum Zweck des Dokuments. Zum Beispiel können in Quelltext Kontrollbefehle fett dargestellt werden, während Daten und Kommentare andere Farben als der Rest des Textes bekommen. Dies verbessert die Lesbarkeit des Textes erheblich und verhilft damit dem Autor zu mehr Effizienz und Produktivität.

```
Theme Repository::defaultTheme(Repository::DefaultTheme t)
{
    if (t == DarkTheme)
        return theme(QLatin1String("Breeze Dark"));
    return theme(QLatin1String("Default"));
}
```

Eine C++-Funktion, mit Hervorhebungen angezeigt.

```
Theme Repository::defaultTheme(Repository::DefaultTheme t)
{
    if (t == DarkTheme)
        return theme(QLatin1String("Breeze Dark"));
    return theme(QLatin1String("Default"));
}
```

Dieselbe C++-Funktion, ohne Hervorhebungen.

Welche der beiden ist einfacher zu lesen?

KatePart enthält ein flexibles, konfigurierbares und leistungsfähiges System für Syntaxhervorhebungen, und die Standarddistribution enthält bereits Definitionen für eine Anzahl von Programmiersprachen, Markup- und Skriptsprachen sowie andere Textformaten. Außerdem können Sie eigene Definitionen in einfachen XML-Dateien erstellen.

KatePart erkennt auf Basis des MIME-Typs, der Dateiendung oder des Inhalts des Dokuments bereits beim Öffnen des Dokuments automatisch die richtigen Regeln für die Syntaxhervorhebungen. Wenn die automatische Auswahl nicht die richtigen Regeln ausgewählt hat, können Sie dies manuell korrigieren (**Extras** → **Hervorhebung**).

Die Schriftstile und Farben, die von jeder Syntaxhervorhebungsdefinition benutzt werden, können auf der Seite Hervorhebungs-Schriftarten des Einrichtungsdialogs festgelegt werden, die Einrichtung der MIME-Typen und Dateierweiterung, auf die diese angewendet werden, ist auf der Seite Dateitypen möglich.

> ANMERKUNG
> Syntaxhervorhebungen sind dazu gedacht die Lesbarkeit von Text zu verbessern, aber nicht dazu geeignet die Richtigkeit des Quelltextes zu überprüfen. Die Erstellung der Regeln für die Hervorhebungen ist kompliziert, abhängig davon, welches Format Sie benutzen. In manchen Fällen sind die Autoren der Regeln stolz, wenn 98 % des Textes korrekt hervorgehoben werden, meistens jedoch sehen Sie die nicht korrekten 2 % nur bei seltenen Konstruktionen.

6.2.2 Das KatePart Syntaxhervorhebungssystem

Dieser Abschnitt behandelt die Mechanismen des KatePart Syntax-Hervorhebungssystems genauer. Wenn Sie selbst Definitionen erstellen oder verändern möchten, sollten Sie diesen genau lesen.

6.2.2.1 Wie es funktioniert

Immer, wenn Sie ein Dokument öffnen, ist eines der ersten Dinge, die KatePart macht, festzustellen, welche Syntaxdefinition für dieses Dokument benutzt werden soll. Während Sie den Text lesen und neuen Text eingeben, analysiert das Syntaxhervorhebungssystem den Text anhand der Regeln in der Syntaxdefinition und markiert ihn dementsprechend.

Wenn Sie Text eingeben, wird der neue Text sofort analysiert und markiert.

Die Syntaxdefinitionen, die in XML benutzt werden, sind XML-Dateien, die Folgendes enthalten

- Regeln für das Erkennen von Text, organisiert in Kontextblöcken
- Listen mit Schlüsselworten
- Stildefinitionen

Beim Analysieren von Text werden die Erkennungsregeln in der Reihenfolge, in der sie definiert wurden, überprüft und wenn der Anfang des aktuellen Textes mit einer Definition übereinstimmt, wird der zugehörige Kontext benutzt. Der nächste Startpunkt wird nach dem Ende des erkannten Bereichs gesetzt und von dort aus wird eine neue Schleife für die Regeln mit dem Kontext der gerade gefundenen Regel gestartet.

6.2.2.2 Regeln

Die Erkennungsregeln sind das Herzstück des Syntaxhervorhebungssystems. Eine Regel besteht aus einer Zeichenfolge, einem Zeichen oder einem regulären Ausdruck. Mit diesen wird der zu analysierende Text verglichen. Sie enthalten Informationen, welche Darstellung für das erkannte Stück Text verwendet werden soll und ob entweder zu einem explizit angegebenem Kontext oder zum vorher vom Text benutzten Kontext gewechselt werden soll.

Die Regeln sind in Kontextgruppen organisiert. Eine Kontextgruppe wird für die grundlegenden Textkonzepte innerhalb des Formates benutzt, z. B. für Textteile in Anführungszeichen oder Kommentarblöcke in Programmquelltext. Dadurch wird sichergestellt, dass sich das Syntaxhervorhebungssystem nicht unnötig durch alle Regeln hindurch arbeiten muss und dass einige Zeichenfolgen im Text abhängig vom aktuellen Kontext unterschiedlich behandelt werden können.

Kontexte können dynamisch generiert werden, um das Benutzen von Daten in Regeln zu erlauben, die nur auf diese Instanz zutreffen.

6.2.2.3 Kontextstile und Schlüsselwörter

In einigen Programmiersprachen werden Ganze Zahlen durch den Compiler (das Programm, das den Quelltext in ein ausführbares Programm übersetzt) anders behandelt als Gleitkommazahlen, und es gibt Zeichen, die eine spezielle Bedeutung innerhalb einer in Anführungszeichen eingeschlossenen Zeichenfolge haben. In solchen Fällen ist es sinnvoll, diese unterschiedlich darzustellen, sodass sie beim Lesen einfach vom umgebenden Text zu unterscheiden sind. Auch wenn diese keine speziellen Kontexte repräsentieren, können sie durch das Syntaxhervorhebungssystem erkannt und anders dargestellt werden.

Eine Syntaxdefinition kann so viele verschiedene Stile beinhalten, wie für das Format notwendig sind.

In vielen Formaten gibt es Listen mit Wörtern, die einem speziellen Konzept zugehörig sind. In Programmiersprachen sind z. B. die Kontrollanweisungen ein Konzept, die Datentypen ein anderes und die eingebauten Funktionen ein drittes. Das Syntaxhervorhebungssystem von KatePart kann benutzt werden, um solche Wörter anhand der Listen zu finden und zur Hervorhebung der Konzepte im Text zu markieren.

6.2.2.4 Standardstile

Wenn Sie eine C++-Quelltextdatei, eine Java™-Quelltextdatei und eine HTML-Datei in KatePart öffnen, sehen Sie dass auch in unterschiedlichen Formaten und damit unterschiedlichen Worten, die spezielle Behandlung bekommen, die benutzten Farben dieselben sind. Der Grund dafür ist, dass KatePart vordefinierte Standardstile benutzt, die von den individuellen Syntaxdefinitionen verwendet werden.

Dadurch wird die Erkennung von ähnlichen Konzepten in verschiedenen Textformaten einfach. Kommentare z. B. gibt es in fast allen Programmiersprachen, Skripten und Markup-Sprachen; diese werden in allen Sprachen gleich dargestellt, sodass Sie sich auf die Arbeit konzentrieren können und nicht über den Zweck einzelner Einträge nachdenken müssen.

TIP
Alle Stile in einer Syntaxdefinition nutzen einen der Standardstile. Einige wenige Syntaxdefinitionen nutzen mehr Stile als Standardstile vorhanden sind. Wenn Sie ein Format sehr oft benutzen, kann es die Arbeit wert sein, den Einrichtungsdialog zu starten und nachzusehen, ob mehrere Konzepte dieselben Stile benutzen. In der Programmiersprache Perl z. B. gibt es zwei Typen von Zeichenfolgen, sodass Sie die Hervorhebung durch eine etwas andere Darstellung des zweiten Typs verbessern können. Alle verfügbaren Standardstile, werden weiter unten erklärt.

6.2.3 Die Hervorhebungsdefinition für das XML Format

6.2.3.1 Überblick

KatePart verwendet die Syntaxhervorhebungs-Bibliothek von KDE Frameworks. Die in KatePart enthaltenen Standard-XML-Hervorhebungsdateien werden in die Syntaxhervorhebungs-Bibliothek einkompiliert.

Dieser Abschnitt ist ein Überblick über die Hervorhebungsdefinition für das XML-Format.. Es beschreibt die Hauptbestandteile, deren Bedeutung und Verwendung. Im nächsten Kapitel werden die Erkennungsregeln detailliert beschrieben.

Die formale Definition XSD finden Sie im Syntax-Highlighting-Repository in der Datei `language.xsd`.

Eigene `.xml`-Dateien mit Definitionen zur Syntaxhervorhebung sind im Ordner `org.kde.syntax-highlighting/syntax/` in Ihrem persönlichen Ordner. Den Pfad zu diesem Ordner finden Sie mit **`qtpaths--paths GenericDataLocation`**. Normalerweise sind dies `$HOME /.local/share/` und `/usr/share/` .

Bei Flatpak- und Snap-Paketen funktioniert der obige Ordner nicht, da der Speicherort der Daten für jede Anwendung unterschiedlich ist. In einer Flatpak-Anwendung ist der Speicherort der benutzerdefinierten XML-Dateien normalerweise `$HOME /.var/app/ flatpak-package-name /data/org.kde.syntax-highlighting/syntax/` und in einer Snap-Anwendung ist dieser Ort `$HOME /snap/ snap-package-name /current/.local/share/org.kde.syntax-highlighting/syntax/` .

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\syntax`. Dabei ist *`%USERPROFILE%`* normalerweise `C:\Users\user`.

Zusammenfassend lässt sich sagen, dass bei den meisten Einrichtungen der Ordner der benutzerdefinierten XML-Dateien wie folgt aussieht

Für lokale Benutzer	`$HOME /.local/share/org.kde.syntax-highlighting/syntax/`
Für alle Benutzer	`/usr/share/org.kde.syntax-highlighting/syntax/`
Für Flatpak-Pakete	`$HOME /.var/app/ flatpak-package-name /data/org.kde.syntax-highlighting/syntax/`
Für Snap-Pakete	`$HOME /snap/ snap-package-name /current/.local/share/org.kde.syntax-highlighting/syntax/`
Unter Windows®	`%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\syntax`

Wenn mehrere Dateien für dieselbe Sprache existieren, wird die Datei mit der höchsten **`version`**-Attribut im **`language`**-Element geladen.

HAUPTBESTANDTEILE DER KATEPART-HERVORHEBUNGSDEFINITIONEN

Eine Hervorhebungsdefinitionsdatei enthält einen Kopf mit der XML-Version:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Die Wurzel der Definitionsdatei ist das Element `language`. Verfügbare Eigenschaften sind:

Notwendige Eigenschaften:

`name` setzt den Namen der Sprache. Dieser erscheint nachher in Menüs und in Dialogen.

Die Eigenschaft **`section`** definiert die Kategorie.

`extensions` definiert die Erweiterungen für Dateinamen wie z. B. "*.cpp;*.h".

`version` gibt die aktuelle Revision der Definitionsdatei als ganze Zahl an. Bei jeder Änderung einer Hervorhebungs-Datei sollte diese Zahl vergrößert werden.

`kateversion` definiert die letzte unterstützte Version von KatePart.

Optionale Eigenschaften:

`mimetype` ordnet Dateien basierend auf deren MIME-Type zu.

`casesensitive` definiert, ob bei den Schlüsselwörtern Groß-/Kleinschreibung unterschieden wird oder nicht.

`priority` ist notwendig, wenn eine andere Hervorhebungsdefinitionsdatei die gleichen Dateinamenerweiterung benutzt. Die Definitionsdatei mit der höheren Priorität wird dann benutzt.

`author` enthält den Namen des Autors und dessen E-Mail-Adresse.

`license` enthält die Lizenz der Datei, normalerweise wird hier die MIT-Lizenz für neue Dateien benutzt.

`style` enthält die Programmiersprache, die mit der Definition zur Verfügung gestellt wird und wird durch das Einrückungsskript für die Eigenschaft `required-syntax-style` benutzt.

`indenter` definiert die als Standard verwendetet Einrückung. Verfügbare Einrückungen sind: *ada*, *normal*, *cstyle*, *cmake*, *haskell*, *latex*, *lilypond*, *lisp*, *lua*, *pascal*, *python*, *replicode*, *ruby* und *xml*.

`hidden` definiert, ob der Name in Menüs von KatePart erscheinen soll.

Die nächste Zeile könnte wie folgt aussehen:

```
<language name="C++" version="1" kateversion="2.4" section="Sources" ↩
    extensions="*.cpp;*.h" />
```

Als nächstes kommt das Element `highlighting`, das das optionale Element `list` und die notwendigen Elemente `contexts` und `itemDatas` enthält.

`list`-Elemente enthalten eine Liste von Schlüsselwörtern. In diesem Fall sind die Schlüsselwörter *class* und *const*. Sie können so viele hinzufügen, wie Sie brauchen.

Seit KDE Frameworks 5.53 kann eine Liste Schlüsselwörter aus anderen Listen oder Sprachen bzw. Dateien enthalten. Dazu benutzen Sie das Element **`include`**. **`##`** wird auf die gleiche Art wie die Regel **`IncludeRules`** verwendet, um den Namen der Liste und der Sprachdefinition zu trennen. Dies ist nützlich, um doppelte Listen von Schlüsselwörtern zu vermeiden, wenn Sie Schlüsselwörter aus anderen Sprachen oder Dateien einschließen müssen. Die Liste *othername* zum Beispiel enthält das Schlüsselwort *str* und alle Schlüsselwörter der Liste *types* aus der Sprache *ISO C++*.

Das Element **`contexts`** enthält alle Kontexte. Der erste Kontext ist Standard bei Start der Hervorhebungen. Es gibt zwei Regeln im Kontext *Normal Text*, die auf die Liste mit Schlüsselwörtern mit dem Namen *somename* und eine Regel, die Anführungszeichen entdeckt und zum Kontext *string* umschaltet. Weitere Informationen zu Regeln finden Sie im nächsten Kapitel.

Der dritte Teil ist das Element **`itemDatas`**. Es enthält alle Farb- und Schriftartstile, die durch die Kontexte und Regeln benötigt werden. In diesem Beispiel werden **`itemData`**, *Normal Text*, *String* und *Keyword* benutzt.

```
<highlighting>
    <list name="somename">
      <item>class</item>
      <item>const</item>
    </list>
    <list name="othername">
      <item>str</item>
      <include>types##ISO C++</include>
    </list>
    <contexts>
      <context attribute="Normal Text" lineEndContext="#pop" name=" ↩
          Normal Text" >
        <keyword attribute="Keyword" context="#stay" String="somename"  ↩
            />
        <keyword attribute="Keyword" context="#stay" String="othername" ↩
             />
        <DetectChar attribute="String" context="string" char="&quot;"  ↩
            />
      </context>
      <context attribute="String" lineEndContext="#stay" name="string"  ↩
          >
        <DetectChar attribute="String" context="#pop" char="&quot;" />
      </context>
    </contexts>
    <itemDatas>
      <itemData name="Normal Text" defStyleNum="dsNormal" />
      <itemData name="Keyword" defStyleNum="dsKeyword" />
      <itemData name="String" defStyleNum="dsString" />
    </itemDatas>
  </highlighting>
```

Der letzte Teil der Hervorhebungsdefinition ist der optionale Abschnitt `general`. Dieser kann Informationen über Schlüsselwörter, Quelltextausblendungen, Leerzeilen und Rechtschreibprüfung enthalten.

Der Abschnitt **`comment`** definiert, mit welcher Zeichenfolge eine einzelne Kommentarzeile beginnt. Sie können außerdem mehrzeilige Kommentare definieren, indem Sie *multiLine* mit der zusätzlichen Eigenschaft *end* benutzen. Diese werden benutzt, wenn Sie das Tastaturkürzel für *Kommentar / Kommentar entfernen* drücken.

Der Abschnitt **`keywords`** definiert, ob in den Schlüsselwortlisten nach Groß- und Kleinschreibung unterschieden wird oder nicht. Andere Eigenschaften werden später erläutert.

Die anderen Abschnitte **`Quelltextausblendung`**, **`Leerzeilen`** und **`Rechtschreibprüfung`** sind normalerweise nicht nötig und werden später erklärt.

```
<general>
    <comments>
      <comment name="singleLine" start="#"/>
      <comment name="multiLine" start="###" end="###" region=" ↩
          CommentFolding"/>
    </comments>
    <keywords casesensitive="1"/>
    <folding indentationsensitive="0"/>
    <emptyLines>
      <emptyLine regexpr="\s+"/>
      <emptyLine regexpr="\s*#.*"/>
    </emptyLines>
    <spellchecking>
      <encoding char="á" string="\'a"/>
```

```
      <encoding char="à" string="\`a"/>
    </spellchecking>
  </general>
</language>
```

6.2.3.2 Die Abschnitte im Einzelnen

Dieser Teil beschreibt alle verfügbaren Eigenschaften für Kontexte, itemDatas, Schlüsselwörter, Kommentare, Quelltextausblendungen und Einrückungen.

Das Element `context` gehört in die Gruppe `contexts`. Ein Kontext selbst definiert spezielle Regeln, wie zum Beispiel, was geschehen soll, wenn das Hervorhebungssystem ein Zeilenende erreicht. Die verfügbaren Eigenschaften sind:

Der Kontextname **`name`**. Regeln benutzen diesen Namen, um festzulegen, zu welchem Kontext umgeschaltet wird, wenn die Regel zutrifft.

Der Kontext **`lineEndContext`** definiert den Kontext, zu dem das Hervorhebungssystem umschaltet, wenn es ein Zeilenende erreicht. Das kann entweder der Name eines anderen Kontextes sein, **`#stay`** um den Kontext nicht umzuschalten, (z. B. tue nichts) oder **`#pop`** das bewirkt, dass der Kontext verlassen wird. Es ist möglich, zum Beispiel **`#pop#pop#pop`** zu verwenden, um drei Kontextebenen zu verlassen oder mit **`#pop#pop!OtherContext`** zwei Kontextebenen zu verlassen und in einen neuen Kontext zu springen. Es ist auch möglich zu einem Kontext zu wechseln, der zu einer anderen Sprachdefinition gehört, genauso wie in den **`IncludeRules`**-Regeln, z. B. **`SomeContext##JavaScript`**. Beachten Sie, dass es nicht möglich ist, diesen Kontextwechsel in Kombination mit **`#pop`** zu verwenden, zum Beispiel ist **`#pop!SomeContext##JavaScript`** nicht gültig. Kontextwechsel werden auch in Abschnitt 6.2.4 beschrieben.

`lineEmptyContext` definiert den Kontext, der in einer leeren Zeile verwendet wird. Die Bezeichnung der Kontextwechsel ist die gleiche wie zuvor in *lineEndContext* beschrieben. Standard hierfür ist: #stay.

`fallthroughContext` legt den nächsten Kontext fest, zu dem gewechselt wird, wenn keine Regel passt. Die Bezeichnung der Kontextwechsel ist die gleiche wie zuvor in *lineEndContext* beschrieben. Voreinstellung: #stay.

`fallthrough` definiert, ob das Hervorhebungssystem zu dem in **`fallthroughContext`** angegebenen Kontext wechselt, wenn keine Regel passt. Beachten Sie, dass seit KDE Frameworks 5.62 dieses Attribut zugunsten von **`fallthroughContext`** veraltet ist. Denn wenn das Attribut **`fallthroughContext`** vorhanden ist, wird stillschweigend angenommen, dass der Wert von **`fallthrough`** *true* ist. Voreinstellung: *false*.

`noIndentationBasedFolding` deaktiviert das auf der Einrückung basierte Ausblenden von Text im Kontext. Wenn das Ausblenden nicht aktiviert ist, ist dieses Attribut nutzlos. Es wird im Element *folding* der Gruppe *general* definiert. Voreinstellung: *false*.

Das Element `itemData` ist in der Gruppe `itemDatas`. Es definiert die Schriftarten und Schriftfarben. So ist es möglich, Ihre eigenen Schriftarten und -farben festzulegen. Wir empfehlen jedoch, bei den vordefinierten Einstellungen zu bleiben, sodass in unterschiedlichen Sprachen trotzdem die gleichen Farben angezeigt werden. Manchmal ist es doch nötig, die Farben und Schriftarten zu ändern. Der Name der Eigenschaft und defStyleNum müssen angeben werden, alle anderen können verwendet werden, sind aber nicht unbedingt nötig. Die verfügbaren Eigenschaften sind:

`name` setzt den Namen von itemData. Kontexte und Regel benutzen diesen Namen in ihrer Eigenschaft *attribute*, um den Bezug zum itemData herzustellen.

`defStyleNum` definiert, welcher Stil standardmäßig benutzt wird. Die verfügbaren Stile werden später näher erläutert.

color definiert eine Farbe. Erlaubte Formate hierfür sind: ‚#rrggbb‘ oder ‚#rgb‘.

selColor definiert die Farbe für die Hervorhebung.

italic Wenn *true*, dann wird der Text in Kursivschrift dargestellt.

bold Wenn *true*, dann wird der Text in Fettschrift dargestellt.

underline Wenn *true*, dann wird der Text unterstrichen dargestellt.

strikeout Wenn *true*, dann wird der Text durchgestrichen dargestellt.

spellChecking Wenn *true*, dann wird die Rechtschreibprüfung für den Text aktiviert.

Das Element `keywords` in der Gruppe `general` definiert Eigenschaften von Schlüsselwörtern. Verfügbare Eigenschaften sind:

casesensitive kann *true* oder *false* sein. Wenn es *true* ist, dann wird bei allen Schlüsselwörtern die Groß- und Kleinschreibung beachtet.

weakDeliminator ist eine Liste von Zeichen, die nicht als Wortbegrenzung wirken. Der Punkt `'.'` ist zum Beispiel eine Wortbegrenzung. Nehmen Sie an, ein Schlüsselwort in einer **list** enthält einen Punkt, diese Schlüsselwort kann nur dann erkannt werden, wenn Sie den Punkt als **weakDeliminator** festlegen.

additionalDeliminator definiert zusätzliche Wortbegrenzungen.

wordWrapDeliminator definiert Zeichen, nach denen ein Zeilenumbruch erfolgen kann.

Standard für Wortbegrenzer und Zeilenumbruchbegrenzer sind die Zeichen `.():!+,-<=>%&*/;?[]^{|}~\`, Leerzeichen (`' '`) und der Tabulator (`'\t'`).

Das Element `comment` in der Gruppe `comments` definiert Eigenschaften für Kommentare, die für Extras → Kommentar, Extras → Kommentar entfernen und Tools → Kommentar ein-/ausschalten benutzt werden. Verfügbare Eigenschaften hierfür sind:

name ist entweder *singleLine* oder *multiLine*. Wenn Sie *multiLine* auswählen, müssen auch die Eigenschaften *end* und *region* benutzt werden. Bei *singleLine* können Sie das optionale Attribut *position* hinzufügen.

start definiert die Zeichenfolge, die einen Kommentar beginnt. In C++ ist dies zum Beispiel *"/*"* in mehrzeiligen Kommentaren. Dieses Attribut ist für *multiLine* und *singleLine* nötig.

end definiert die Zeichenfolge, die einen Kommentar beendet. In C++ ist dies zum Beispiel *"*/"*. Diese Attribut ist nur für den Typ *multiLine* verfügbar und erforderlich.

region sollte der Name von ausblendbaren Mehrzeilenkommentaren sein. Nehmen Sie an, Sie haben *beginRegion=Comment ... endRegion=Comment* in Ihren Regeln, dann sollten Sie *region=Comment* benutzen. Auf diesem Wege funktioniert das automatische Entfernen von Kommentaren auch dann, wenn Sie nicht den gesamten Text des mehrzeiligen Kommentars auswählen. Es muss nur der Cursor innerhalb des mehrzeiligen Kommentars stehen. diese Attribut ist nur für den Typ *multiLine* verfügbar.

position definiert, wo der einzeilige Kommentar eingefügt wird. Standardmäßig wird der einzeilige Kommentar am Anfang der der Zeile bei Spalte 0 platziert, aber wenn Sie *position="afterwhitespace"* verwenden, wird der Kommentar nach führenden Leerraumzeichen rechts eingefügt, vor dem ersten Nicht-Leerraumzeichen. Dies ist nützlich für das korrekte Einfügen von Kommentaren in Sprachen, in denen die Einrückung wichtig ist, wie z. B. in Python oder YAML. Dieses Attribut ist optional und der einzig mögliche Wert ist *afterwhitespace*. Dies ist nur verfügbar für den Typ *singleLine*.

Das Element `folding` in der Gruppe `general` definiert Eigenschaften für ausblendbaren Quelltext. Verfügbare Eigenschaften sind:

indentationsensitive Wenn *true*, werden die Markierungen für Quelltextausblendungen basiert auf Einrückungen gesetzt, wie zum Beispiel in der Skriptsprache Python. Normalerweise brauchen Sie dies nicht zu setzen, Standard ist *false*.

Das Element `emptyLine` in der Gruppe `emptyLines` definiert, welche Zeilen als Leerzeilen behandelt werden sollen. Damit lässt sich das Verhalten des Attributs *lineEmptyContext* in den Elementen `Kontext` ändern. Verfügbare Attribute sind:

`regexpr` definiert einen regulären Ausdruck, der als eine leere Zeile behandelt wird. Standardmäßig enthalten leere Zeilen keine Zeichen, daher werden hier zusätzliche Leerzeilen hinzugefügt, z. B. wenn Zeilen mit Leerzeichen als Leerzeilen betrachtet werden sollen. In den meisten Syntaxdefinitionen brauchen Sie dieses Attribut jedoch nicht zu setzen.

Das Element `encoding` in der Gruppe `spellchecking` definiert eine Zeichenkodierung für die Rechtschreibprüfung. Verfügbare Eigenschaften sind:

`char` ist ein kodiertes Zeichen.
`string` ist eine Folge von Zeichen, die in der Rechtschreibprüfung als das Zeichen *char* kodiert wird. In der Sprache LaTeX repräsentiert beispielsweise die Zeichenfolge **`\¨{A}`** das Zeichen **Ä**.

6.2.3.3 Verfügbare Standardstile

Standardstile wurden als kurze Zusammenfassung bereits erklärt. Standardstile sind vordefinierte Schriftarten und -farben.

Allgemeine Standardstile:

`dsNormal`, wenn keine spezielle Hervorhebung benötigt wird

`dsKeyword`, benutzt für eingebaute Sprach-Schlüsselwörter.

`dsFunction`, benutzt für Funktionsaufrufe und -definitionen.

`dsVariable`, falls zutreffend Variablennamen z. B. $someVar in PHP/Perl.

`dsControlFlow`, Kontrollfluss-Schlüsselwörter wie if, else, switch, break, return, yield, ...

`dsOperator`, Operatoren wie + - * / :: < >

`dsBuiltIn`, eingebaute Funktionen, Klassen und Objekte.

`dsExtension`, allgemeine Erweiterungen wie zum Beispiel Qt™-Klassen und Funktionen/Makros in C++ und Python.

`dsPreprocessor`, Präprozessor-Anweisungen oder Makro-Definitionen.

`dsAttribute`, Anmerkungen wie @override und __declspec(...).

Standardstile für Zeichenfolgen:

`dsChar`, benutzt für einzelne Buchstaben wie „X“.

`dsSpecialChar`, Zeichen mit besonderer Bedeutung in Zeichenfolgen wie Escape-Sequenzen, Ersetzungen oder Operatoren für reguläre Ausdrücke.

`dsString`, benutzt für Zeichenfolgen wie „Hallo Welt“.

`dsVerbatimString`, wörtliche oder unveränderte Zeichenfolgen wie „raw \backlash“ in Perl, CoffeeScript und Shells wie auch r'\raw' in Python.

`dsSpecialString`, SQL, Reguläre Ausdrücke, HERE-Dokumente, LaTeX-Mathematikmodus, ...

`dsImport`, import, include, erforderliche Module.

Standardstile für Zahlen:

`dsDataType`, benutzt für eingebaute Datentypen wie int, void, u64.

`dsDecVal`, benutzt für Dezimalwerte.

`dsBaseN`, benutzt für Werte mit einer anderen Zahlenbasis als 10.

`dsFloat`, benutzt für Gleitkommawerte.

`dsConstant`, eingebaute und benutzerdefinierte Konsonanten wie Pi.PI.

Standardstile für Kommentare und Dokumentation:

dsComment, benutzt für Kommentare.

dsDocumentation, /** Dokumentation-Kommentare */ oder """docstrings""".

dsAnnotation, Dokumentations--Befehle wie @param, @brief.

dsCommentVar, die in den vorher genannten Befehlen verwendeten Variablennamen wie „foobar“ in @param foobar.

dsRegionMarker, benutzt für Markierungen von Bereichen wie //BEGIN, //END in Kommentaren.

Andere Standardstile:

dsInformation, Notizen und Hinweise wie @note in doxygen.

dsWarning, Warnungen wie @warning in doxygen.

dsAlert, besondere Wörter wie TODO, FIXME, XXXX.

dsError, benutzt für Hervorhebungen von Fehlern und für fehlerhafter Syntax.

dsOthers, wenn nichts anderes passt.

6.2.4 Hervorhebungs-Erkennungsregeln

Dieser Abschnitt beschreibt die Hervorhebungs-Erkennungsregeln

Jede Regel kann auf Null oder mehrere Zeichen am Anfang der untersuchten Zeichenfolge zutreffen. Wenn eine Übereinstimmung gefunden wird, wird den erkannten Zeichen der Stil oder die *Eigenschaft*, die durch die Regel festgelegt wurde, zugeordnet, Außerdem kann die Regel ein Umschalten des aktuellen Kontexts anfordern.

Eine Regel sieht wie folgt aus:

```
<RuleName attribute="(identifier)" context="(identifier)" [rule specific ↩
    attributes] />
```

Die *attribute* (Eigenschaft) legt den Namen des Stils fest, der für die erkannten Zeichen benutzt werden soll und der *context* (Kontext) legt den Kontext fest, der ab hier benutzt werden soll.

Der *context* (Kontext) kann durch Folgendes identifiziert werden:

- Einen *identifier*, der der Name eines anderen Kontextes ist.
- Eine Anweisung, die vorgibt, im aktuellen Kontext zu bleiben (**#stay**), oder zu einem vorher in der Zeichenfolge benutzten Kontext zurückzuspringen (**#pop**).

  Zum Zurückgehen über mehrere Schritte kann das Schlüsselwort #pop wiederholt werden: **#pop#pop#pop**
- Eine Anweisung *order*, die von einem Ausrufezeichen (*!*) und einem *identifier* gefolgt wird, veranlasst Kate erst die Anweisung *order* auszuführen und dann in den anderen Kontext umzuschalten, z. B. **#pop#pop!OtherContext**.
- Ein *identifier* ist ein Kontextname gefolgt von zwei Doppelkreuzen (**##**) und einem weiteren *identifier* für den Name einer Sprachdefinition. Diese Namensgebung ist ähnlich wie bei den Regeln **IncludeRules** und ermöglicht den Wechsel zu einem Kontext, der zu einer anderen Syntaxhervorhebungsdefinition gehört, z. B. **SomeContext##JavaScript**. Beachten Sie, dass es nicht möglich ist, diesen Kontextwechsel in Kombination mit **#pop** zu verwenden, z. B. **#pop!SomeContext##JavaScript** ist nicht gültig.

Regelspezifische Eigenschaften sind unterschiedlich und werden im Folgenden beschrieben.

GEMEINSAME EIGENSCHAFTEN

- *attribute*: Eine Eigenschaft zeigt auf ein bestimmtes *itemData*-Element.
- *context*: Legt den Kontext fest, zu dem das Hervorhebungssystem umschaltet, wenn die Regel als zutreffend erkannt wird.
- *beginRegion*: Beginnt einen Quelltextausblendungsblock. Standard ist: unset.
- *endRegion*: Beendet eine Quelltextausblendungsblock. Standard ist: unset.
- *lookAhead*: Wenn *true*, dann wird das Hervorhebungssystem die Länge der Übereinstimmung nicht verarbeiten. Standard ist: *false*.
- *firstNonSpace*: Trifft nur dann zu, wenn die Zeichenfolge als erstes nach Zwischenräumen in der Zeile erkannt wird. Standard ist: *false*.
- *column*: Trifft nur dann zu, wenn die Spalte zutrifft. Standard ist: unset.

Dynamische Regeln

- *dynamic*: kann *(true oder false)* sein.

Wie es funktioniert:
In den regulären Ausdrücken der der **RegExpr**-Regeln wird der gesamte Text innerhalb einfacher runder Klammern **(PATTERN)** erfasst und behalten. Diese Erfassungen können in dem Kontext verwendet werden, in den gewechselt wird, in den Regeln mit dem Attribut **dynamic** *true*, durch `%N` (in *String*) oder `N` (in *char*).

Ein Text, der in einer **RegExpr**-Regel erfasst wird, nur für den gewechselten Kontext behalten wird, der in seinem Attribut **Kontext** angegeben ist.

> **TIP**
>
> - Wenn die Erfassung nicht verwendet werden sollen, sowohl durch dynamische Regeln als auch im gleichen regulären Ausdruck, sollte **nicht-erfassende Gruppen** verwendet werden verwendet werden: **(?:PATTERN)**
>
>   Die Gruppen *Vorwärtsreferenz* oder *Rückwärtsreferenz* wie **(?=PATTERN)**, **(?!PATTERN)** oder **(?<=PATTERN)** werden nicht erfasst. Weitere Informationen fingen Sie im Abschnitt Reguläre Ausdrücke.
> - Die Erfassungs-Gruppen können innerhalb desselben regulären Ausdrucks verwendet werden, indem `\N` anstelle von `%N` verwendet wird. Für weitere Informationen siehe Erfassen von passendem Text (Rückwärtsreferenz) in AbschnittReguläre Ausdrücke.

Beispiel 1:

In diesem einfachen Beispiel wird der Text, der mit dem regulären Ausdruck **=*** übereinstimmt, erfasst und für `%1` in die dynamische Regel eingefügt. Dadurch kann der Kommentar mit der gleiche Zahl von Gleichheitszeichen **=** wie am Anfang beendet werden. Dies passt auf Text wie: **[[Kommentar]]**, **[=[Kommentar]=]** oder **[=====[Kommentar]=====]**.

Außerdem sind die Erfassungen nur im gewechselten Kontext *mehrzeiligen Kommentaren* verfügbar.

```
<context name="Normal" attribute="Normal Text" lineEndContext="#stay">
  <RegExpr context="Multi-line Comment" attribute="Comment" String="\[(=*) ↩
     \[" beginRegion="RegionComment"/>
</context>
<context name="Multi-line Comment" attribute="Comment" lineEndContext="# ↩
    stay">
  <StringDetect context="#pop" attribute="Comment" String="]%1]" dynamic=" ↩
     true" endRegion="RegionComment"/>
</context>
```

Beispiel 2:

In der dynamischen Regel entspricht *%1* der Erfassung, die auf **#+** passt und *%2* auf **"+**. Dies trifft auf Text wie **#label˝˝˝˝inside the context˝˝˝˝#** zu.

Diese Erfassungen sind in anderen Kontexten wie z. B. *OtherContext*, *FindEscapes* oder *SomeContext* nicht verfügbar.

```
<context name="SomeContext" attribute="Normal Text" lineEndContext="#stay">
  <RegExpr context="#pop!NamedString" attribute="String" String="(#+)(?:[\w ↩
     -]|[^[:ascii:]])(&quot;+)"/>
</context>
<context name="NamedString" attribute="String" lineEndContext="#stay">
  <RegExpr context="#pop!OtherContext" attribute="String" String="%2(?:%1) ↩
     ?" dynamic="true"/>
  <DetectChar context="FindEscapes" attribute="Escape" char="\"/>
</context>
```

Beispiel 3:

Die passt auf Text wie: **Class::function<T>(...)**.

```
<context name="Normal" attribute="Normal Text" lineEndContext="#stay">
  <RegExpr context="FunctionName" lookAhead="true"
              String="\b([a-zA-Z_][\w-]*)(::)([a-zA-Z_][\w-]*)(?:&lt;[\w\-\ ↩
                 s]*&gt;)?(\()"/>
</context>
<context name="FunctionName" attribute="Normal Text" lineEndContext="#pop">
  <StringDetect context="#stay" attribute="Class" String="%1" dynamic="true ↩
     "/>
  <StringDetect context="#stay" attribute="Operator" String="%2" dynamic=" ↩
     true"/>
  <StringDetect context="#stay" attribute="Function" String="%3" dynamic=" ↩
     true"/>
  <DetectChar context="#pop" attribute="Normal Text" char="4" dynamic="true ↩
     "/>
</context>
```

LOKALE BEGRENZUNGSZEICHEN

- *weakDeliminator*: Liste der Zeichen, die nicht als Wortbegrenzungen fungieren.
- *additionalDeliminator* definiert zusätzliche Wortbegrenzungen.

6.2.4.1 Die Regeln im Einzelnen:

DetectChar

Findet ein einzelnes bestimmtes Zeichen. Häufig zum Finden des Endes von Zeichenfolgen in Anführungszeichen benutzt.

```
<DetectChar char="(character)" (common attributes) (dynamic) />
```

Die Eigenschaft **char** definiert das zu erkennende Zeichen.

Detect2Chars

Findet zwei bestimmte Zeichen in einer bestimmten Reihenfolge.

```
<Detect2Chars char="(character)" char1="(character)" (common attributes ↩
    ) />
```

Die Eigenschaft **char** definiert das erste zu erkennende Zeichen, **char1** das zweite.

AnyChar

Findet ein Zeichen aus einem bestimmten Satz von Zeichen.

```
<AnyChar String="(string)" (common attributes) />
```

Die Eigenschaft **String** definiert den Satz der Zeichen.

StringDetect

Findet eine bestimmte Zeichenfolge.

```
<StringDetect String="(string)" [insensitive="true|false"] (common ↩
    attributes) (dynamic) />
```

Die Eigenschaft **String** definiert die zu erkennende Zeichenfolge. Die Eigenschaft **insensitive** ist standardmäßig auf *false* gesetzt und wird an die Zeichenfolgen-Vergleichsfunktion übergeben. Wenn der Wert auf *true* gesetzt wird, wird Groß- und Kleinschreibung ignoriert.

WordDetect

Findet eine Zeichenfolge, aber zusätzlich werden die Wortgrenzen wie ein Punkt `'.'` oder ein Leerzeichen am Anfang und Ende des Wortes beachtet. Dies funktioniert wie der reguläre Ausdruck **\b<string>\b**, ist aber schneller als die Regel **RegExpr**.

```
<WordDetect String="(string)" [insensitive="true|false"] (common ↩
    attributes) (local deliminators) />
```

Die Eigenschaft **String** definiert die zu erkennende Zeichenfolge. Die Eigenschaft **insensitive** ist standardmäßig auf *false* gesetzt und wird an die Zeichenfolgen-Vergleichsfunktion übergeben. Wenn der Wert auf *true* gesetzt wird, wird Groß- und Kleinschreibung ignoriert.

Ab Version: Kate 3.5 (KDE 4.5)

RegExpr

Prüft die Übereinstimmung mit einem regulären Ausdruck.

```
<RegExpr String="(string)" [insensitive="true|false"] [minimal="true| ↩
    false"] (common attributes) (dynamic) />
```

Die Eigenschaft **String** definiert den regulären Ausdruck.

Die Eigenschaft **insensitive** ist standardmäßig auf *false* gesetzt und wird an die Funktion zur Auswertung des regulären Ausdrucks übergeben.

Die Eigenschaft **minimal** ist standardmäßig auf *false* gesetzt und wird an die Funktion zur Auswertung des regulären Ausdrucks übergeben.

Weil die Regeleinhaltung immer am Anfang der aktuellen Zeichenfolge geprüft wird, kann mit dem Hochzeichen (^) angegeben werden, dass die Regeleinhaltung nur am Anfang der Zeile untersucht werden soll.
Sehen Sie unter Reguläre Ausdrücke für weitere Informationen zu diesen nach.

keyword

Erkennt ein Schlüsselwort aus einer angegebenen Liste.

```
<keyword String="(list name)" (common attributes) (local deliminators) ↩
    />
```

Die Eigenschaft **String** definiert die Schlüsselwortliste durch deren Name. Eine Liste mit diesem Namen muss vorhanden sein.

Das Hervorhebungssystem verarbeitet die Regeln mit sehr stark optimierten Methoden. Deswegen ist es absolut notwendig, dass alle Schlüsselworte, die gefunden werden sollen, durch definierte Begrenzer eingeschlossen werden. Das können entweder die Standardbegrenzer sein oder Begrenzer, die mit der Eigenschaft *additionalDeliminator* des Tags *keywords* festgelegt wurden.

Wenn ein Schlüsselwort ein Begrenzerzeichen enthalten soll, dann muss dieses Zeichen zur Eigenschaft *weakDeliminator* des Tags *keywords* hinzugefügt werden. Dieses Zeichen verliert damit seine Funktion als Begrenzer in allen *keyword*-Regeln. Es ist auch möglich, das *weakDeliminator* Attribut vom *keyword* zu verwenden, so dass diese Änderung nur für diese Regel gilt.

Int

Erkennt eine ganze Zahl wie im regulären Ausdruck **\b[0−9]+**).

```
<Int (common attributes) (local deliminators) />
```

Diese Regel hat keine speziellen Eigenschaften.

Float

Erkennt eine Dezimalzahl wie im regulären Ausdruck **\b[0−9]+)\.[0−9]*|\.][−+]?[0−9]+)?**).

```
<Float (common attributes) (local deliminators) />
```

Diese Regel hat keine speziellen Eigenschaften.

HlCOct

Erkennt eine Oktalzahl wie im regulären Ausdruck **\b0[0−7]+**.

```
<HlCOct (common attributes) (local deliminators) />
```

Diese Regel hat keine speziellen Eigenschaften.

HlCHex

Erkennt eine hexadezimale Zahl wie im regulären Ausdruck **\b0[xX][0−9a−fA−F]+**.

```
<HlCHex (common attributes) (local deliminators) />
```

Diese Regel hat keine speziellen Eigenschaften.

HlCStringChar

Findet ein Steuerzeichen.

```
<HlCStringChar (common attributes) />
```

Diese Regel hat keine speziellen Eigenschaften.

Solche Zeichen sind durch druckbare Zeichen dargestellte nicht druckbare Zeichen, die in Programmquelltexten häufig benutzt werden. z. B.: **\n** (Zeilenvorschub) oder **\t** (TAB)

Die folgenden Zeichen werden erkannt, wenn sie einem Linksschrägstrich \ folgen: **abef nrtv"'?**. Zusätzlich werden auch hexadezimale (**\xff**) oder oktale (**\033**) Zahlen nach einem \ erkannt.

HlCChar

Findet ein C Zeichen.

```
<HlCChar (common attributes) />
```

Diese Regel hat keine speziellen Eigenschaften.

Trifft zu, wenn C Zeichen in einfachen Anführungszeichen (Beispiel: **'c'**) vorkommen. In den Anführungszeichen kann ein einfaches Zeichen oder Sonderzeichen (Beispiel: **' '**) stehen. Für Zeichenfolgen von Sonderzeichen sehen Sie unter HlCStringChar nach.

RangeDetect

Findet eine Zeichenfolge mit definierten Anfangs- und Endzeichen.

```
<RangeDetect char="(character)"  char1="(character)" (common attributes ↩
    ) />
```

char definiert das Zeichen am Anfang des Bereichs, **char1** das Zeichen am Ende des Bereichs.
Diese Regel ist für das Finden von kleinen Zeichenfolgen in Anführungszeichen nützlich, kann aber wegen der verwendeten Funktion keine über mehrere Zeilen gehenden Zeichenfolgen finden.

LineContinue

Trifft auf ein angegebenes Zeichen an einem Zeilenende zu.

```
<LineContinue (common attributes) [char="\"] />
```

Die Eigenschaft **char** definiert das optionale zu erkennende Zeichen, Standard ist der Rückstrich **'\'**. Neu seit KDE 4.13.
Diese Regel wird zum Umschalten des Kontextes am Ende einer Zeile benutzt. Dies wird in C/C++ zum Fortsetzen von Makros oder Zeichenfolgen gebraucht.

IncludeRules

Schließt Regeln aus einem anderen Kontext, einer anderen Sprache oder einer anderen Datei ein.

```
<IncludeRules context="contextlink" [includeAttrib="true|false"] />
```

Die Eigenschaft **context** definiert, welcher Kontext eingeschlossen werden soll.

Wenn dies eine einfache Zeichenfolge ist, dann werden alle definierten Regeln in den gegenwärtigen Kontext eingeschlossen. Beispiel:

```
<IncludeRules context="anotherContext" />
```

Wenn die Zeichenfolge eine **##**-Nutzereingabe enthät, dann wird das Hervorhebungssystem einen Kontext aus einer anderen Sprachdefinition mit dem angegebenen Namen suchen, zum Beispiel:

```
<IncludeRules context="String##C++" />
```

schliesst den Kontext *String* aus der Sprachdefinition für *C++* ein.

Wenn die Eigenschaft **includeAttrib** *true* ist, dann wird die Zieleigenschaft zu der aus der Quelle geändert. Dies wird zum Beispiel für Kommentare gebraucht, wenn der Text, der durch den eingeschlossenen Kontext anders hervorgehoben wird, als im gegenwärtigen Kontext.

DetectSpaces

Finde Zwischenräume.

```
<DetectSpaces (common attributes) />
```

Diese Regel hat keine speziellen Eigenschaften.

Benutzen Sie diese Regel, wenn Sie wissen, dass jetzt mehrere Zwischenräume folgen, zum Beispiel am Anfang von eingerückten Zeilen. Diese Regel überspringt mehrere Zwischenräume mit einem Mal, ohne diese einzeln auf die Einhaltung von anderen Regeln zu testen und dann nach Nichtzutreffen einzeln zu überspringen.

DetectIdentifier

Finde Zeichenfolgen als Bezeichner (als regulärer Ausdruck: **`[a-zA-Z_][a-zA-Z0-9_]*`**).

```
<DetectIdentifier (common attributes) />
```

Diese Regel hat keine speziellen Eigenschaften.

Benutzen Sie diese Regel zum Überspringen von Wörtern mit einem Mal, ohne die Zeichen im Wort einzeln auf die Einhaltung von anderen Regeln zu testen und dann nach Nichtzutreffen zu überspringen.

6.2.4.2 Tipps & Tricks

- Wenn Sie nur zwei Zeichen vergleichen, dann benutzen Sie **`Detect2Chars`** an Stelle von **`StringDetect`**. Das Gleiche gilt für **`DetectChar`**.
- Reguläre Ausdrücke sind einfach zu benutzen, aber oft gibt es einen anderen viel schnelleren Weg, um das gleiche Ergebnis zu erreichen. Nehmen Sie an, Sie wollen feststellen, ob das Zeichen **`'#'`** das erste Zeichen einer Zeile ist. Ein regulärer Ausdruck dafür wäre:

```
<RegExpr attribute="Macro" context="macro" String="^\s*#" />
```

Sie können aber auch die wesentlich schnellere Lösung:

```
<DetectChar attribute="Macro" context="macro" char="#" firstNonSpace=" ↩
    true" />
```

benutzen. An Stelle des regulären Ausdrucks **`'^#'`** können Sie **`DetectChar`** mit der Eigenschaft **`column=˝0˝`** benutzen. Die Eigenschaft **`column`** zählt Zeichenbasiert, sodass auch ein Tabulator nur ein Zeichen ist.

- Verwenden Sie in **`RegExpr`**-Regeln das Attribut **`column=˝0˝`**, wenn mit dem Muster **`^PATTERN`** Text am Anfang einer Zeile gefunden werden soll. Dies ist schneller, da nicht mehr in den restlichen Spalten der Zeile nach Übereinstimmungen gesucht wird.
- Verwenden Sie in regulären Ausdrücken nicht-erfassende Gruppen **`(?:PATTERN)`** anstelle von erfassenden Gruppen **`(PATTERN)`**, wenn die Erfassungen nicht in demselben regulären Ausdruck oder in dynamischen Regeln verwendet werden. Dadurch wird das unnötige Speichern von Erfassungen vermieden.
- Sie können zwischen Kontexten umschalten, ohne Zeichen zu verarbeiten. Angenommen, Sie wollen den Kontext umschalten, wenn Sie die Zeichenfolge **`*/`** finden, aber Sie müssen diese Zeichenfolge im nächsten Kontext verarbeiten. Die folgende Regel trifft zu und die Eigenschaft **`lookAhead`** sorgt dafür, dass die zutreffende Zeichenfolge für den folgenden Kontext bereitgehalten wird.

```
<Detect2Chars attribute="Comment" context="#pop" char="*" char1="/"  ↩
    lookAhead="true" />
```

- Benutzen Sie **`DetectSpaces`**, wenn Sie wissen, dass mehrere Zwischenräume vorkommen.
- Benutzen Sie **`DetectIdentifier`** an Stelle des regulären Ausdrucks **`'[a-zA-Z_]\w*'`**.

- Benutzen Sie Standardstile wann immer das möglich ist. Die Benutzer finden dadurch eine vertraute Umgebung vor.
- Sehen Sie in anderen XML-Dateien nach, wie andere Benutzer komplizierte Regeln geschrieben haben.
- Sie können die Gültigkeit jeder XML-Datei mit dem Befehl **validatehl.sh language.xsd mySyntax.xml** überprüfen. Die Dateien `validatehl.sh` und `language.xsd` finden Sie im Syntax-Highlighting-Repository.
- Wenn Sie komplexe reguläre Ausdrücke oft wiederholen, können Sie *ENTITIES* benutzen. Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE language SYSTEM "language.dtd"
[
        <!ENTITY myref    "[A-Za-z_:][\w.:_-]*">
]>
```

  Nun können Sie *&myref;* an Stelle des regulären Ausdrucks benutzen.

6.3 Arbeiten mit Farbschemata

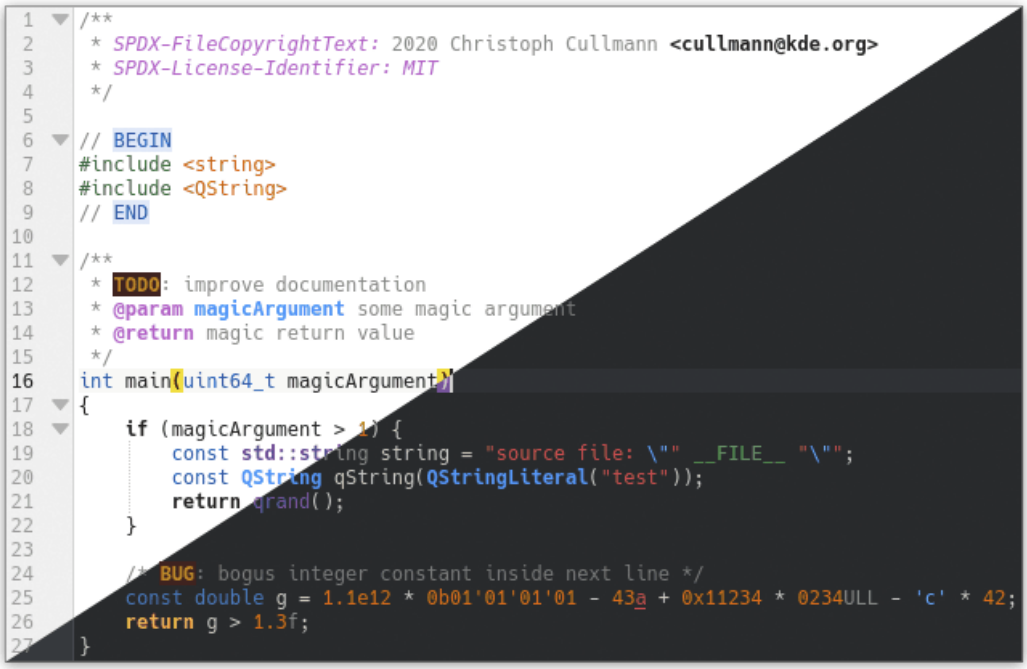
6.3.1 Überblick

Farbschemata definieren die Farben des Text-Editierbereichs und der Syntaxhervorhebung. Ein Farbschema umfasst die folgenden Punkte:

- Der Textstil für die Syntaxhervorhebung durch die *Standard-Stilattribute*. Zum Beispiel die Textfarbe und die ausgewählte Textfarbe.
- Der Hintergrund des Textbearbeitungsbereichs, einschließlich der Textauswahl und der aktuellen Zeile.
- Die Symbolumrandung des Textbereichs, deren Hintergrund, die Trennlinie, die Zeilennummern, die Zeilenumbruchmarkierungen, die geänderten Zeilenmarkierungen und die Quelltextausblendung.
- Textmarkierungen wie die Suchmarkierungen, die Einrückung und die Zeilenmarkierungen für Tabulator-/Leerzeichen, zusammengehörende Klammern und die Rechtschreibprüfung.
- Lesezeichen und Textbausteine.

Um Verwechslungen zu vermeiden, liegt das Folgende außerhalb des Anwendungsbereichs:

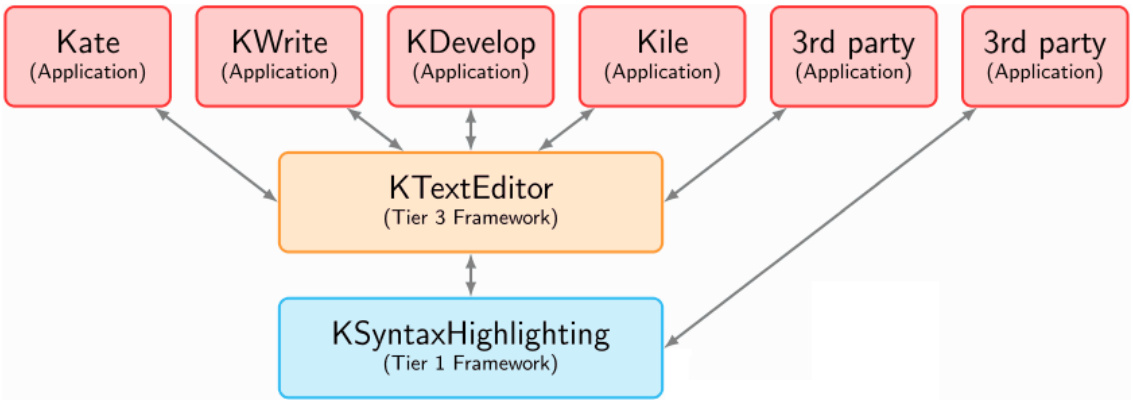
- Die Schriftart und Schriftgröße.
- Die Farben der Textbearbeitung, wie z. B. die Textgrafik auf der Bildlaufleiste, die Menüs, die Unterfensterleiste, die Fensterfarbe usw. In KDE Anwendungen, wie Kate oder KDevelop, werden diese Farben durch das **`globale KDE Plasma,Farbschema`** definiert, die im Modul Farben in den Systemeinstellungen oder von der Anwendung selbst im Menü **Einstellungen** → **Farbschema** festgelegt werden.



Die Farbschemata „Breeze-Hell" und „Breeze-Dunkel" mit der Syntaxhervorhebung für „C++".

6.3.2 Farbschemata für KSyntaxHighlighting

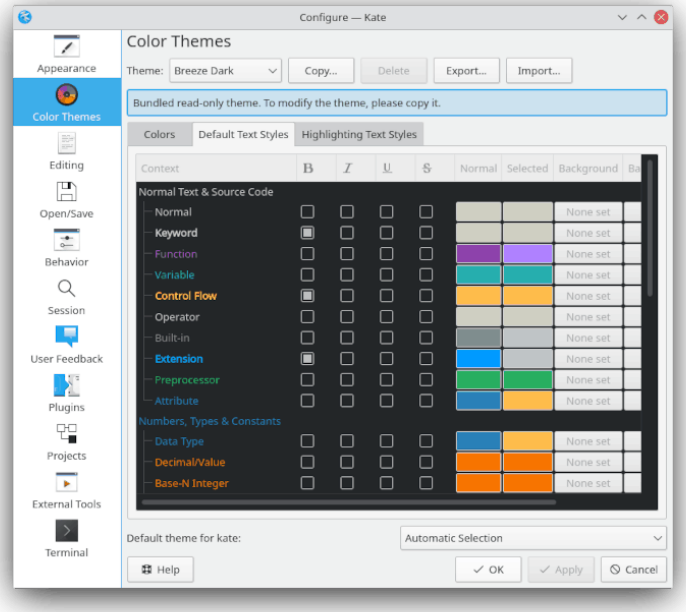
Das Framework KSyntaxHighlighting enthält die Syntax-Highlighting-Engine und ist eine Bibliothek, die **die Farbschemata bereitstellt und verwaltet**. Die Bibliothek ist Teil von von KDE-Frameworks und wird in KDE-Texteditoren wie Kate, KWrite, Kile und KDevelop verwendet. Diese Abhängigkeit sieht wie folgt aus:



Abhängigkeit von KDE-Frameworks-Bibliotheken von Texteditoren.

KSyntaxHighlighting enthält eine Vielzahl von eingebauten Schemata, die auf der Seite Farbschemata der Webseite des Kate-Editors zu finden sind.

Das Framework KTextEditor enthält eine Benutzeroberfläche zum Erstellen und Bearbeiten von Farbschemata und ermöglicht das Importieren und Exportieren von Schemata. Sie können diesen Dialog in den „Einstellungen" des Texteditors öffnen. Weitere Informationen finden Sie im Abschnitt Abschnitt 6.3.5.



Die GUI zur Bearbeitung von Farbschemata in den Einstellungen von Kate.

In den KDE-Texteditoren wie Kate oder KDevelop werden die Farbschemata von KSyntaxHighlighting seit KDE Frameworks 5.75 vom 10. Oktober 2020 verwendet. Zuvor wurden die Farbschemata von Kate (KConfig-basierte Schema-Einstellungen) verwendet und die sind nun veraltet. Es ist jedoch möglich, die alten Kate-Schemata in die KSyntaxHighlighting-Farbschemata umzuwandeln. Das KSyntaxHighlighting-Quelltextarchiv enthält das Skript **`utils/katescheme_to_theme_converter.py`** und den Ordner **`utils/schema-converter/`** mit weiteren Hilfsprogrammen für diesen Zweck.

6.3.3 Das JSON-Format der Farbschemata

6.3.3.1 Überblick

Farbschemata werden in Dateien im JSON-Format mit der Dateierweiterung **`.theme`** gespeichert.

Im KSyntaxHighlighting-Quelltext befinden sich die JSON-Dateien der eingebauten Schemata im Ordner **`data/themes/`** . Die eingebauten Schemata werden in die Bibliothek KSyntaxHighlighting kompiliert, daher ist der Zugriff auf sie über den Quelltext oder durch Exportieren aus der GUI zur Verwaltung der Schemata von KTextEditor möglich.

Sie können auch auf einfache Weise zusätzliche oder benutzerdefinierte Schemata hinzuzufügen, die aus dem Dateisystem geladen werden. Benutzerdefinierte Schemadateien befinden sich im Ordner `org.kde.syntax-highlighting/themes/` in Ihrem persönlichen Ordner, den Sie mit dem Befehl **`qtpaths --paths GenericDataLocation`** finden können, üblicherweise `$HOME /.local/share/` und `/usr/share/` .

Bei Flatpak- und Snap-Paketen funktioniert der obige Ordner nicht, da der Speicherort der Daten für jede Anwendung unterschiedlich ist. In einer Flatpak-Anwendung ist der Speicherort der benutzerdefinierten Schemadateien normalerweise `$HOME /.var/app/ flatpak-package-name /data/org.kde.syntax-highlighting/themes/` und in einer Snap-Anwendung ist dieser Ort `$HOME /snap/ snap-package-name /current/.local/share/org.kde.syntax-highlighting/themes/` .

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\themes`. *`%USERPROFILE%`*. Dabei ist *`%USERPROFILE%`* normalerweise `C:\Users\user-name`.

Zusammenfassend lässt sich sagen, dass der Ordner für benutzerdefinierte Schemata bei den meisten Einrichtungen wie folgt aussieht:

Für lokale Benutzer	`$HOME /.local/share/org.kde.syntax-highlighting/themes/`
Für alle Benutzer	`/usr/share/org.kde.syntax-highlighting/themes/`
Für Flatpak-Pakete	`$HOME /.var/app/ flatpak-package-name /data/org.kde.syntax-highlighting/themes/`
Für Snap-Pakete	`$HOME /snap/ snap-package-name /current/.local/share/org.kde.syntax-highlighting/themes/`
Unter Windows®	`%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\themes`

Wenn mehrere Schemadateien mit demselben Namen existieren, wird die Datei mit der höchsten **`Revision`** geladen.

6.3.3.2 Dit JSON-Struktur

Der Aufbau einer JSON-Datei wird auf dieser Webseite erläutert. Grundsätzlich besteht eine Datei im JSON-Format aus:

- Sammlungen von Schlüssel/Wert-Paaren, getrennt durch Kommas und gruppiert in **`{ }`**, auch „Objekte" genannt.
- Sortierte Listen von Werten, getrennt durch Kommas und gruppiert in **`[ ]`**, auch „Feld" genannt.

Die Bezeichnungen „Schlüssel", „Wert", „Objekt" und „Feld" werden hier verwendet. Falls Sie zum ersten Mal mit JSON-Dateien arbeiten, helfen die folgenden Beispiele beim Verständnis.

6.3.3.3 Hauptabschnitte der JSON-Farbschemadateien

Das Basisobjekt der Farbschemadateien im JSON-Format enthält die folgenden Schema-Schlüssel:

- **`metadata`**: Dies ist obligatorisch. Der Wert ist ein Objekt mit den Metadaten des Schemas wie Name, Revision und Lizenz.
  Ausführliche Informationen dazu unter Abschnitt 6.3.3.4.
- **`Editor-Farben`**: Dies ist obligatorisch. Der Wert ist ein Objekt mit den Farben des Textbearbeitungsbereichs wie dem Hintergrund, dem Symbolrand und der Textdekoration.
  Ausführliche Informationen dazu unter Abschnitt 6.3.4.1.
- **`Textstile`**: Dies ist obligatorisch. Der Wert ist ein Objekt mit dem Attribut*Standard-Textstil* der Syntaxhervorhebung. Jedes Attribut definiert seine *Textfarbe*, seine *gewählte Textfarbe*, oder ob es zum Beispiel *fett* oder *kursiv* sein soll. Die Textstile können aus den Attributen der Syntaxdefinitionsdateien referenziert werden.
  Ausführliche Informationen dazu unter Abschnitt 6.3.4.2.

- **custom-styles**: Ist optional. Definiert Textstile für die Attribute bestimmter Definitionen von Syntaxhervorhebungen. In einer Hervorhebungsdefinition wie **Python** oder **Markdown** können Sie einen anderen Textstil angeben, der den in **text-styles** definierten Standard überschreibt.
  Ausführliche Informationen dazu unter Abschnitt 6.3.4.3.

Die Sprache JSON unterstützt keine Kommentare. Sie können jedoch mit dem optionalen Schlüssel **_comments** im Basisobjekt Kommentare schreiben. Wenn sie zum Beispiel ein bestehendes Schema anpassen, können Sie die URL des ursprünglichen Archivs angeben. Der praktischste Weg ist die Verwendung eines Felds mit Zeichenfolgen.

Nachfolgend finden Sie eine Beispieldatei für das Schema „Breeze-Hell"-. Um das Beispiel nicht zu lang werden zu lassen, enthalten die Objekte **editor-colors** und **text-styles** nicht alle alle erforderlichen Schlüssel. Sie können die vollständige Datei des Schemas Breeze-Hell aus dem KSyntaxHighlighting-Quelltextarchiv herunterladen.

```
{
    "_comments": [
        "This is a comment.",
        "If this theme is an adaptation of another, put the link to the  ↩
           original repository."
    ],
    "metadata": {
        "name" : "Breeze Light",
        "revision" : 5,
        "copyright": [
            "SPDX-FileCopyrightText: 2016 Volker Krause <vkrause@kde.org>",
            "SPDX-FileCopyrightText: 2016 Dominik Haumann <dhaumann@kde.org ↩
               >"
        ],
        "license": "SPDX-License-Identifier: MIT"
    },
    "editor-colors": {
        "BackgroundColor" : "#ffffff",
        "CodeFolding" : "#94caef",
        "BracketMatching" : "#ffff00",
        "CurrentLine" : "#f8f7f6",
        "IconBorder" : "#f0f0f0",
        "IndentationLine" : "#d2d2d2",
        "LineNumbers" : "#a0a0a0",
        "CurrentLineNumber" : "#1e1e1e",
        The other editor color keys...

    },
    "text-styles": {
        "Normal" : {
            "text-color" : "#1f1c1b",
            "selected-text-color" : "#ffffff",
            "bold" : false,
            "italic" : false,
            "underline" : false,
            "strike-through" : false
        },
        "Keyword" : {
            "text-color" : "#1f1c1b",
            "selected-text-color" : "#ffffff",
            "bold" : true
        },
        "Function" : {
```

```
            "text-color" : "#644a9b",
            "selected-text-color" : "#452886"
        },
        "Variable" : {
            "text-color" : "#0057ae",
            "selected-text-color" : "#00316e"
        },
        The other text style keys...

    },
    "custom-styles": {
        "ISO C++": {
            "Data Type": {
                "bold": true,
                "selected-text-color": "#009183",
                "text-color": "#00b5cf"
            },
            "Keyword": {
                "text-color": "#6431b3"
            }
        },
        "YAML": {
            "Attribute": {
                "selected-text-color": "#00b5cf",
                "text-color": "#00b5cf"
            }
        }
    }
}
```

6.3.3.4 Metadaten

Das JSON Objekt des Schlüssels **`metadata`** enthält die wichtigsten Informationen über das Schema. Dieses Objekt hat die folgenden Schlüssel:

- **`Name`**: Dies ist eine *Zeichenfolge* mit dem Name der Sprache. Dies wird in den Menüs und Dialogen angezeigt und ist verpflichtend.
- **`revision`**: Es ist eine *ganze Zahl*. die die aktuelle Revision der Schemadatei angibt. Wann immer Sie eine Farbschemadatei aktualisieren, müssen Sie diese Zahl vergrößern. Diese Wert ist erforderlich.
- **`Lizenz`**: Eine *Zeichenfolge*, die die Lizenz des Schemas definiert, unter Verwendung des Lizenzbezeichner **`SPDX-License-Identifier`** aus dem Standard SPDX-Lizenzkommunikationsformat. Dies ist optional.

  Die vollständige Liste der SPDX-Lizenzbezeichner können Sie hier einsehen.
- **`copyright`**: Ein *Feld* mit *Zeichenfolgen*, das die Autoren des Schemas angibt, mit **`SPDX-File CopyrightText`** aus dem Standard SPDX-Lizenzkommunikationsformat. Dies ist optional.

```
"metadata": {
    "name" : "Breeze Light",
    "revision" : 5,
    "copyright": [
        "SPDX-FileCopyrightText: 2016 Volker Krause <vkrause@kde.org>",
        "SPDX-FileCopyrightText: 2016 Dominik Haumann <dhaumann@kde.org>"
    ],
```

```
    "license": "SPDX-License-Identifier: MIT"
}
```

6.3.4 Farben im Detail:

In diesem Abschnitt werden all verfügbaren Attribute und Einstellungen für Farben aufgeführt:

6.3.4.1 Editor-Farben

Entspricht den Farben im Editorbereichs.

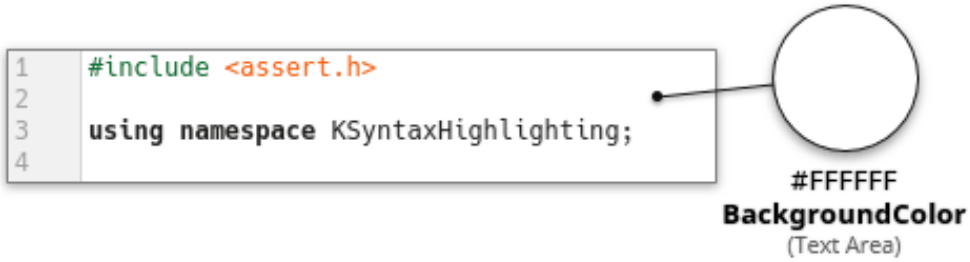
In der JSON-Schemadatei ist der Wert des entsprechenden Schlüssels **`editor-colors`** ein *object* wobei jeder Schlüssel auf eine Attribut-Farbe des des Texteditors verweist. Hier sind **`alle verfügbaren Schlüssel notwendig`**, ihre Werte sind **`Zeichenfolgen`** mit hexadezimalen Farbcodes wie „#00B5CF".

In der GUI zum Verwalten von Schemata für KTextEditor können diese Attribute auf der Karteikarte **`Farben`** ändern.

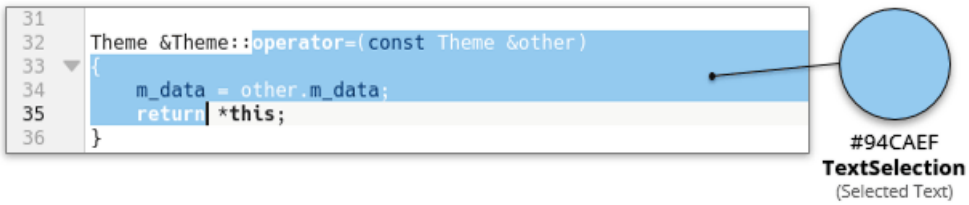
Folgende Schlüssel sind verfügbar: Die Schlüssel aus der JSON-Datei sind *fett*, die Namen in der GUI sind in Klammern angegeben.

Hintergrundfarben des Editors

`BackgroundColor` (Textbereich)
Dies ist die Standardhintergrundfarbe für den Editorbereich, die vorherrschende Farbe im Editorbereich.



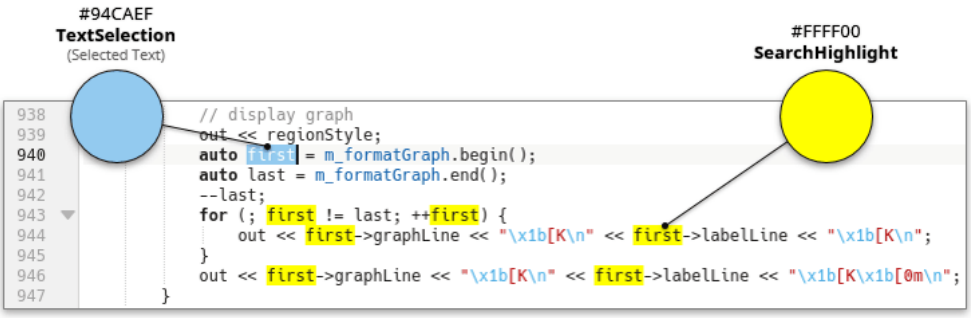
`TextSelection` (Ausgewählter Text)
Dies ist die Hintergrundfarbe für ausgewählten Text.



`CurrentLine` (Aktuelle Zeile)
Setzt die Farbe für die aktuelle Zeile. Die Farbe ist ein klein wenig anders als die normale Hintergrundfarbe, sodass Sie die aktuelle Zeile schnell wiederfinden.

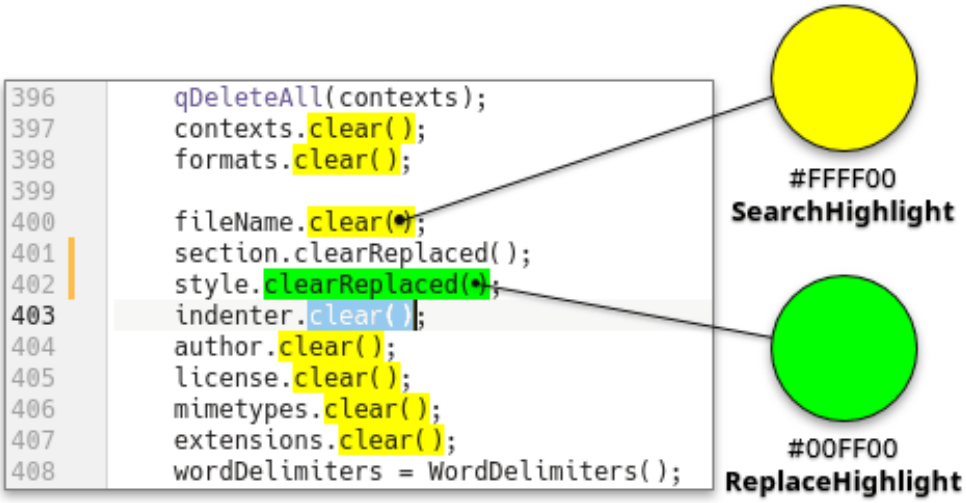
`SearchHighlight` (Suchen-Hervorhebung)

Die Farbe für den Text, der bei der letzten Suche gefunden wurde.



`ReplaceHighlight` (Ersetzen-Hervorhebung)

Die Farbe für den Text, der bei der letzten Suche gefunden wurde.



Symbolrand

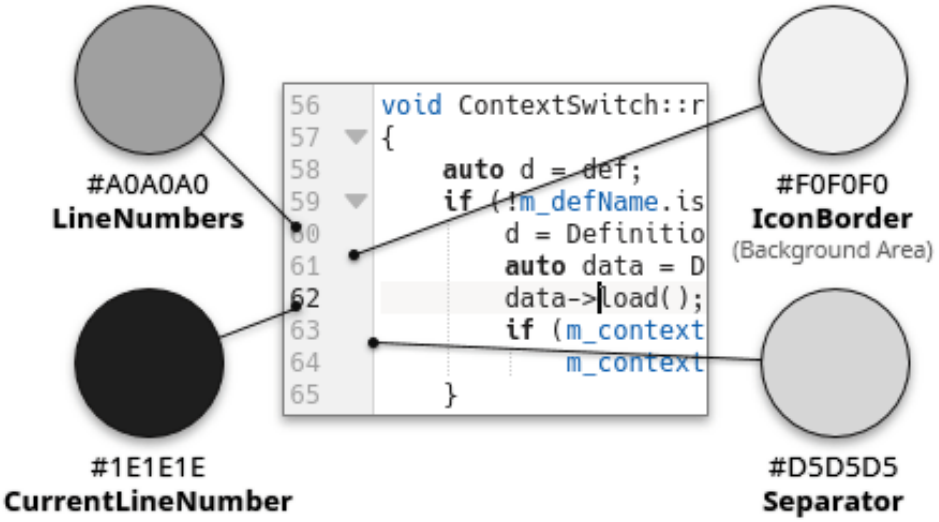
`IconBorder` (Hintergrundbereich)

Diese Farbe wird für den Hintergrund des Symbolrandes und des Zeilennummerrandes an der linken Seite des Editorfensters verwendet.

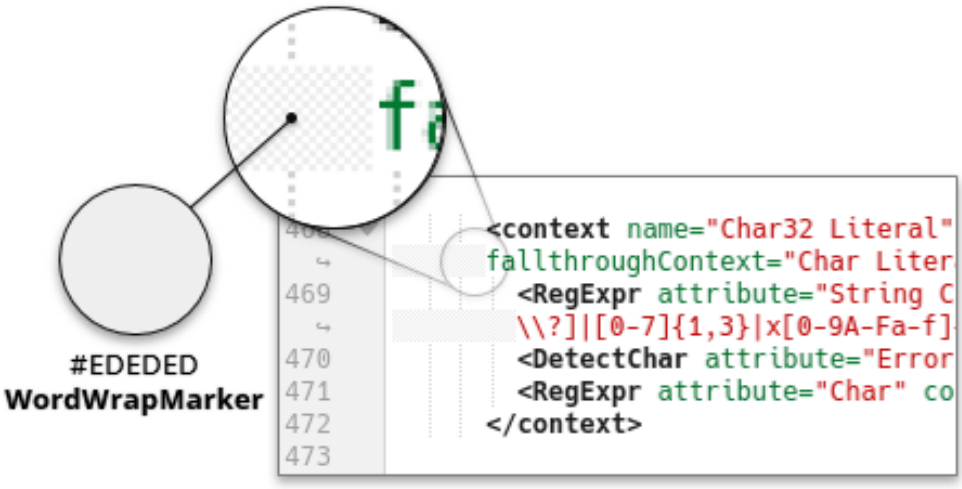
`LineNumbers` (Zeilennummern)
Diese Farbe wird für die Zeilennummern am linken Rand des Editorbereiches verwendet.

`CurrentLineNumber` (Aktuelle Zeilennummer)
Diese Farbe wird verwendet, um die Zeilennummer der aktuellen Zeile darzustellen und wird auf der linken Seite der Ansicht eingeblendet. Wenn Sie dies etwas anders als für„LineNumbers" einstellen, erleichtert das die aktuellen Zeile im Blick zu behalten.

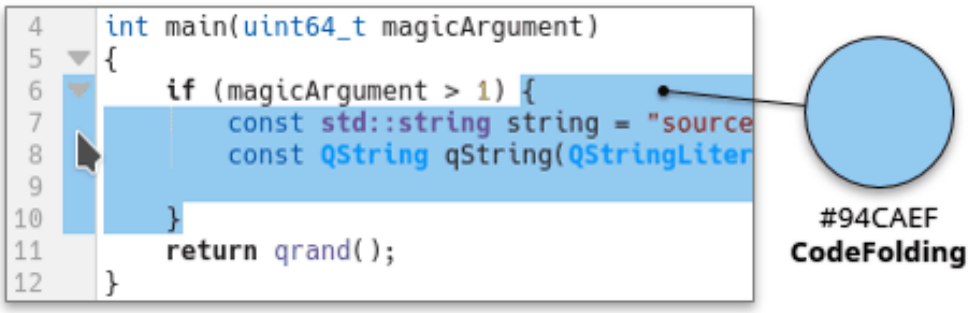
`Separator` (Trennlinie)
Diese Farbe wird verwendet, um die senkrechte Linie zu zeichnen, die den Symbolrand vom Hintergrund des Textbereichs trennt.



`WordWrapMarker` (Markierungen für Zeilenumbrüche)
Diese Farbe wird benutzt, wenn am linken Rand angezeigt wird, dass Zeilen dynamisch umgebrochen und eingerückt sind, sowie auch für die Markierung von festen Zeilenumbrüchen.



`CodeFolding` (Quelltextausblendung)
Mit dieser Farbe wird der Abschnitt des Quelltextes hervorgehoben, der beim Klicken auf den Pfeil zur Quelltextausblendung am linken Rand des Dokuments ausgeblendet wird Weitere Informationen finden Sie im Abschnitt Quelltextausblendung.

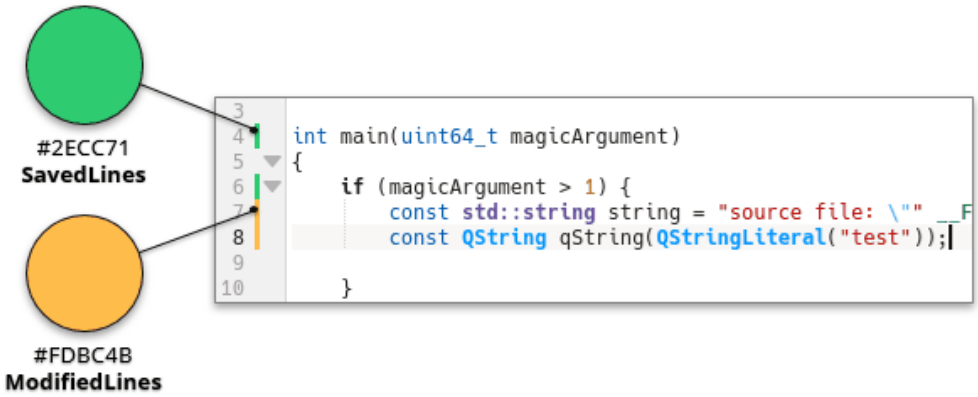


`ModifiedLines` (Geänderte Zeilen)

Mit dieser Farbe werden links neben dem Dokument Zeilen hervorgehoben, die in dieser Sitzung geändert wurden aber noch nicht gespeichert sind. Weitere Informationen finden Sie unter Abschnitt 3.9.

`SavedLines` (Gespeicherte Zeilen)

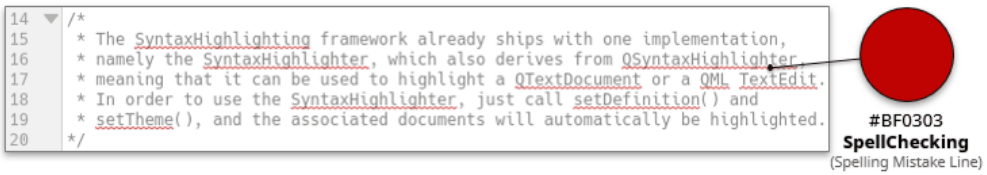
Mit dieser Farbe werden links neben dem Dokument Zeilen hervorgehoben, die in dieser Sitzung geändert wurden und bereits gespeichert sind. Weitere Informationen finden Sie unter Abschnitt 3.9.



Textdekorationen

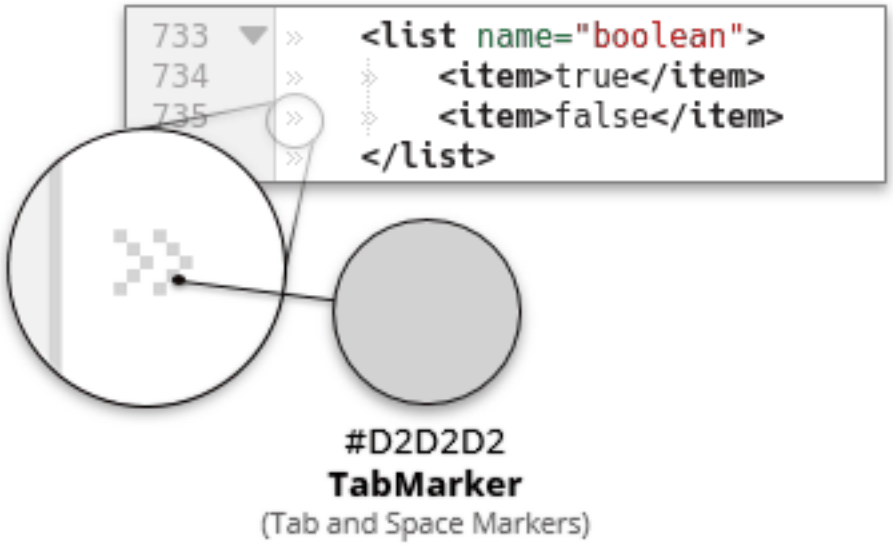
`SpellChecking` (Linie für Rechtschreibfehler)

Dies legt die Farbe der Linie fest, die zum Markieren von Rechtschreibfehlern verwendet wird.



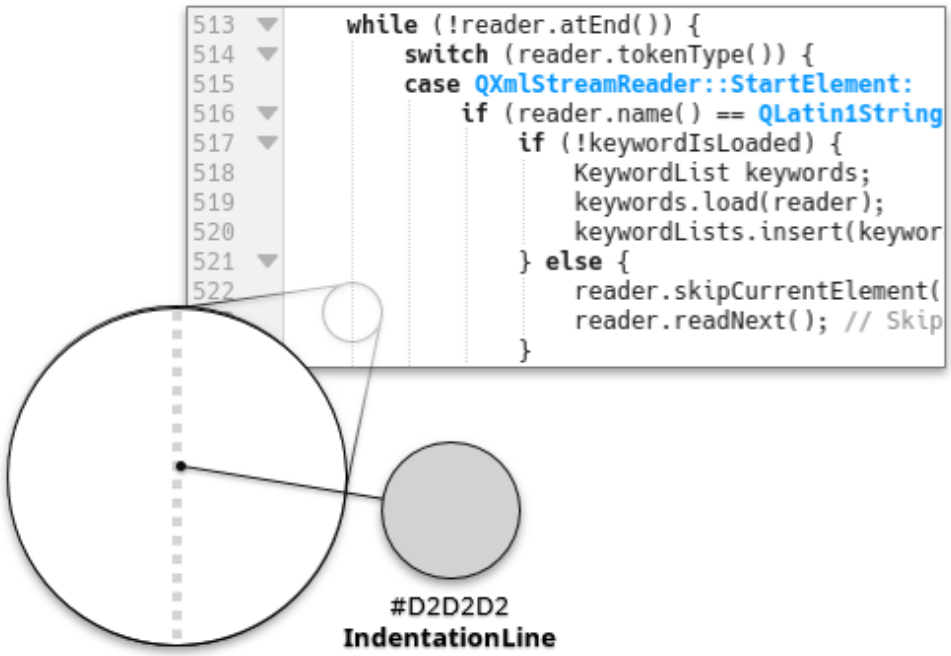
`TabMarker` (Markierungen für Tabulatoren und Leerzeichen)

Diese Farbe wird für dir Markierung von Tabulatoren und Leerzeichen verwendet, wenn Symbole für Wortzwischenräume angezeigt werden.



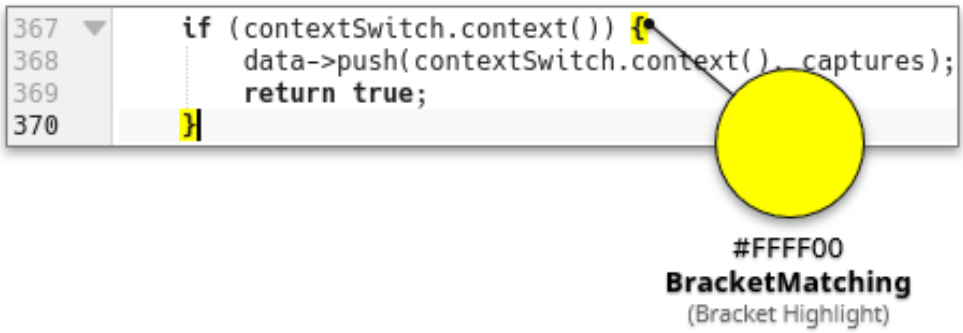
`IndentationLine` (Einrückungslinie)

Diese Farbe wird verwendet,um eine Linie links von eingerückten Textblöcken anzuzeigen, wenn diese Funktion aktiviert ist.



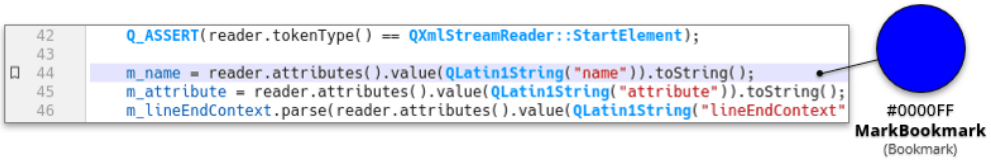
`BracketMatching` (Hervorhebung für Klammern)

Diese Farbe wird für den Hintergrund von zusammengehörenden Klammern verwendet.



Markierungsfarben

`MarkBookmark` (Lesezeichen)
Mit dieser Farbe werden Lesezeichen angezeigt. Beachten Sie, dass diese Farbe eine Deckkraft von 22 % (und 33 % für die aktuelle Zeile) in Bezug auf den Hintergrund hat. Weitere Informationen finden Sie unter Abschnitt 3.6.



`MarkBreakpointActive` (Aktiver Haltepunkt)
In dieser Farbe wird vom GDB-Modul ein aktiver Haltepunkt angezeigt, Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat. Weitere Informationen finden Sie in der Dokumentation zum GDB-Modul.

`MarkBreakpointReached` (Erreichter Haltepunkt)
In dieser Farbe wird vom GDB-Modul ein Haltepunkt angezeigt, der bei der Fehlersuche erreicht wurde. Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat. Weitere Informationen finden Sie in der Dokumentation zum GDB-Modul.

`MarkBreakpointDisabled` (Nicht aktiver Haltepunkt)
In dieser Farbe wird vom GDB-Modul ein nicht aktiver Haltepunkt angezeigt, Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat. Weitere Informationen finden Sie in der Dokumentation zum GDB-Modul.

`MarkExecution` (Ausführung)
In dieser Farbe wird vom GDB-Modul die zur Zeit ausgeführte Zeile angezeigt, Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat Weitere Informationen finden Sie in der Dokumentation zum GDB-Modul.

`MarkWarning` (Warnung)
Mit dieser Farbe wird vom Erstellen-Modul eine Zeile eingefärbt, für die Compiler eine Warnung ausgegeben hat. Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat. Weitere Informationen finden Sie in der Dokumentation zum Erstellen-Modul.

`MarkError` (Fehler)
Mit dieser Farbe wird vom Erstellen-Modul eine Zeile eingefärbt, für die Compiler einen Fehler ausgegeben hat Beachten Sie, dass diese Farbe eine Deckkraft gegenüber dem Hintergrund hat. Weitere Informationen finden Sie in der Dokumentation zum Erstellen-Modul.

Textvorlagen & Textbausteine

`TemplateBackground` (Hintergrund)

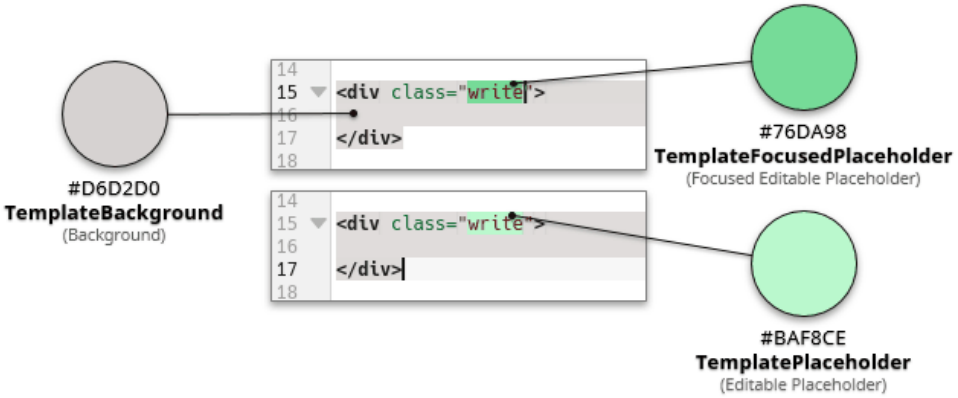
In dieser Farbe wird der Hintergrund eines Textbausteins in Kate angezeigt. Weitere Informationen dazu finden Sie in der Dokumentation zu Kate-Textbausteinen.

`TemplatePlaceholder` (Editierbarer Platzhalter)

In dieser Farbe wird ein Platzhalter eines Textbausteins in Kate angezeigt, den Sie zur manuellen Bearbeitung anklicken können. Weitere Informationen dazu finden Sie in der Dokumentation zu Kate-Textbausteinen.

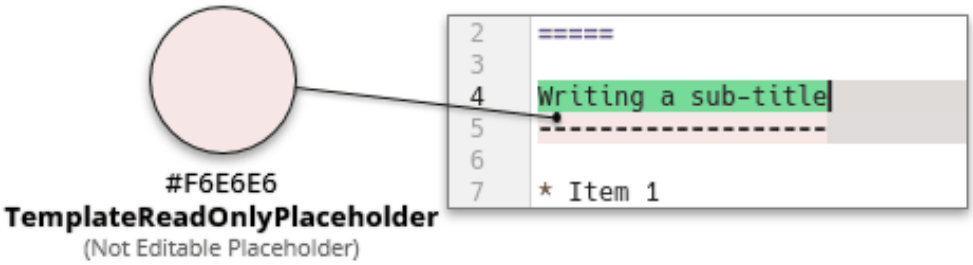
`TemplateFocusedPlaceholder` (Editierbarer Platzhalter mit Fokus)

In dieser Farbe wird der gerade bearbeitete Platzhalter eines Textbausteins in Kate angezeigt Weitere Informationen dazu finden Sie in der Dokumentation zu Kate-Textbausteinen.



`TemplateReadOnlyPlaceholder` (Nicht editierbarer Platzhalter)

Diese Farbe wird im Textbausteinmodul von Kate verwendet, um Platzhalter zu kennzeichnen, der nicht manuell editiert werden kann und zum Beispiel automatisch ausgefüllt wird. Weitere Informationen finden Sie in der Dokumentation zu Kate-Textbausteinen.



6.3.4.2 Standardtextstile

Die Stile für Standardtext sind von den Stilen für Hervorhebungen abgeleitet, sodass der Editor Texte immer in der gleichen Form anzeigen kann. So sind zum Beispiel Kommentare unabhängig vom Textformat oder der Programmiersprache des Quelltextdokuments immer in der gleichen Farbe gekennzeichnet.

> ANMERKUNG
> Diese Textstile können von den **`Standardstilen`** in den Definitionen der XML-Dateien zur Syntaxhervorhebung referenziert werden. Zum Beispiel entspricht das Attribut „Normal" dem Attribut „dsNormal" in den XML-Dateien, und „DataType" entspricht „dsDataType". Weitere Informationen finden Sie in der Abschnitt 6.2.3.3Dokumentation zur Syntaxhervorhebung.

> **TIP**
> Achten Sie darauf, lesbare Farben mit gutem Kontrast zu wählen, besonders in Kombination mit dem **`Editor-Farben`**. Siehe auch Abschnitt 6.3.6.1.

In der JSON-Datei, hat der jeweilige Schlüssel **`text-styles`** als Wert ein *Objekt*, wobei jeder Schlüssel dem Namen eines *Standard-Textstils* in den in den Definitionen für die Syntaxhervorhebung entspricht. Hier sind **`alle verfügbaren Stil-Schlüsselwörter für Text obligatorisch`** , diese sind unten aufgeführt.

```
"text-styles": {
    "Normal" : {
        "text-color" : "#1f1c1b",
        "selected-text-color" : "#ffffff",
        "bold" : false,
        "italic" : false,
        "underline" : false,
        "strike-through" : false
    },
    "Keyword" : {
        "text-color" : "#1f1c1b",
        "selected-text-color" : "#ffffff",
        "bold" : true
    },
    "Function" : {
        "text-color" : "#644a9b",
        "selected-text-color" : "#452886"
    },
    The other text style keys...
}
```

Jeder Schlüssel des *Standardtextstils* enthält ein JSON-Objekt als Wert, in dem Werte wie *color*, *bold*, *italic*, usw. angegeben werden. Es gibt folgende Schlüssel:

`text-color`: Eine *Zeichenfolge* mit der Textfarbe im hexadezimalen Farbcode. Dieser Schlüssel/Wert ist erforderlich.

`selected-text-color`: Die Textfarbe für ausgewählten Text hat in der Regel den gleichen Wert wie „text-color". Wenn der Text ausgewählt ist, wird der Hintergrund durch den Wert von TextSelection in der Editor-Farben definiert, daher sollte der Text einen guten Kontrast haben und mit diesem Hintergrund gut lesbar ist. Der Wert ist eine *Zeichenfolge* mit einem hexadezimalen Farbcode. Dieser Schlüssel/Wert muss angegeben werden.

`bold`: Ein *Boolescher Wert*, der festlegt, ob der Text in Fettschrift angezeigt wird. Dieser Schlüssel ist optional, der Standardwert ist **`false`**.

`italic`: Ein *Boolescher Wert*, der festlegt, ob der Text kursiv angezeigt wird. Dieser Schlüssel ist optional, der Standardwert ist **`false`**.

`underline`: Ein *Boolescher Wert*, der festlegt, ob der Text in unterstrichen angezeigt wird. Dieser Schlüssel ist optional, der Standardwert ist **`false`**.

`strike-through`: Ein *Boolescher Wert*, der festlegt, ob der Text in durchgestrichen angezeigt wird. Dieser Schlüssel ist optional, der Standardwert ist **`false`**.

`background-color`: Bestimmt die Hintergrundfarbe von Text und wird zum Beispiel in Warnungen (Alerts) in Kommentaren verwendet. Der Wert ist eine *Zeichenfolge* mit einem hexadezimalen Farbcode. Dieser Schlüssel ist optional, als Voreinstellung gibt es keine Hintergrundfarbe.

selected-background-color: Bestimmt die Hintergrundfarbe von ausgewähltem Text. Der Wert ist eine *Zeichenfolge* mit einem hexadezimalen Farbcode. Dieser Schlüssel ist optional, als Voreinstellung gibt es keine Hintergrundfarbe.

In der GUI zur Verwaltung der Farbthemen von KTextEditor können diese Attribute auf der Karteikarte **Standardtextstile** geändert werden. Der Name in der Liste der Stile verwendet den für das Element eingerichteten Stil, so dass Sie beim Ändern eines Stils sofort eine Vorschau erhalten. Jeder Stil ermöglicht die Auswahl gemeinsamer Attribute sowie von Vorder- und Hintergrundfarben. Um eine Hintergrundfarbe zu deaktivieren, klicken Sie mit der rechten Maustaste, um das Kontextmenü zu verwenden.
Folgende Schlüssel für Textstile sind verfügbar: Die Schlüssel aus der JSON-Datei sind *fett* geschrieben, die Namen in der GUI sind in Klammern angegeben, falls sie anders sind.

Normaltext & Quelltext

Normal: Standard-Textstil für normalen Text und Quelltext ohne besondere Hervorhebung.

Keyword: Textstil für eingebaute Sprach-Schlüsselwörter.

Function: Textstil für Funktionsaufrufe und -definitionen.
Variable: Textstil für Variablen, falls zutreffend. Zum Beispiel beginnen Variablen in PHP/Perl typischerweise mit einem **$**, also werden alle Bezeichner mit dem Muster **$foo** als Variable hervorgehoben.

ControlFlow (**Kontrollfluss**): Textstile für Schlüsselwörter den Programmablauf wie zum Beispiel *if*, *then*, *else*, *return*, *switch*, *break*, *yield*, *continue*, usw.

Operator: Textstil für Operatoren wie **+**, **−**, *****, **/** , **%** usw.

BuiltIn > (**Built-in**): Textstil eingebaute Funktionen, Klassen und Objekte.

Extension: Textstil für bekannte Erweiterungen wie zum Beispiel Qt™-Klassen und Funktionen/Makros in C++ und Python oder Boost.

PreprocessorTextstil für Präprozessor-Anweisungen oder Makro-Definitionen.

Attribute: Textstil für Anmerkungen oder Attribute von Funktionen oder Objekten wie zum Beispiel **@override** in Java oder **__declspec(...)** und **__attribute__((...))** in C++.

Zahlen, Typen & Konstanten

DataType (**Datentyp**): Textstil für eingebaute Datentypen wie *int*, *char*, *float*, *void*, *u64* usw.

DecVal (**Dezimal/Wert**): Textstil für Dezimalwerte.

BaseN (**Base-N Integer**): Textstil für Werte mit einer anderen Zahlenbasis als 10.

Float (**Fließkommazahl**): Textstil für Gleitkommawerte.

Constant: Textstile für Konstanten in Sprachen oder für benutzerdefinierte Konstanten wie zum Beispiel *True*, *False*, *None* in Python oder *nullptr* in C/C++; oder mathematische Konstanten wie *PI*.

Zeichenfolgen & Zeichen

Char (**Zeichen**): Textstil für einzelne Zeichen wie **'x'** .

SpecialChar (**Sonderzeichen**): Textstil für maskierte Zeichen in Zeichenfolgen, z.B. „**he llo** ", und andere Zeichen mit besonderer Bedeutung in Zeichenfolgen, z. B. Ersetzungen oder Regex-Operatoren.

String:Textstil für Zeichenfolgen wie „**Hallo Welt**".

VerbatimString (**Wörtliche Zeichenfolge**): Textstile für wörtliche oder unveränderte Zeichenfolgen wie **raw \backlash** in Perl, CoffeeScript und Shells wie auch **r'\raw'** in Python oder wie in HERE-Dokumenten.

SpecialString (**Besondere Zeichenfolge**): Textstil für besondere Zeichenfolgen wie reguläre Ausdrücke in ECMAScript, im LaTeX-Mathematikmodus, SQL, usw.

Import (**Importe, Module, Includes**): Textstil für Includes, Importe, Module oder LaTeX-Pakete.

Kommentare & Dokumentation

Comment: Textstile für normale Kommentare.

Documentation: Textstil für Kommentare zur API-Dokumentation wie z. B. **/** doxygen comments */** oder **"""docstrings"""**.

Annotation: Textstil für Anmerkungen in Kommentaren oder Dokumentationsbefehle wie **@param** in Doxygen oder JavaDoc.

KommentarVar (**Kommentar-Variable**): Textstil, der auf Variablennamen verweist, die in obigen Befehlen in einem Kommentar verwendet werden, wie z. B. **foobar** in „**@param foobar**" in Doxygen oder JavaDoc.

RegionMarker (**Bereichsmarkierungr**): Textstil für das Markieren von Bereichen, typischerweise definiert durch**/ /BEGIN** und **/ /END** in Kommentaren.

Information:Textstil für Notizen und Hinweise wie zum Beispiel Schlüsselwörter wie **@note** in Doxygen.

Warning: Textstil für Warnungen wie das Schlüsselwort **@warning** in Doxygen.

Alert: Textstil für spezielle Wörter in Kommentaren wie **TODO**, **FIXME**, **XXXX** und **WARNING**.

Verschiedenes

Error: Textstil für Hervorhebungen von Fehlern und für fehlerhafter Syntax.

Others: Textstile für Attribute, die nicht für einen der anderen Standardstile zutreffen.

6.3.4.3 Benutzerdefinierte Textstile für Hervorhebungen

Hier können Sie Textstile für eine bestimmte Syntaxhervorhebung festlegen Definition festlegen und damit den **Standard-Textstil** überschreiben wie in dem vorherigen Abschnitt beschrieben.

In der JSON-Schemadatei entspricht dies dem Schlüssel **custom-styles**, dessen Wert ein ein *Objekt* ist, wobei jeder Unterschema-Schlüssel mit dem **Name einer Syntaxhervorhebungsdefinition** korrespondiert. Sein Wert ist ein *Objekt*, bei dem jeder Schlüssel auf den **Name des Stilattributs** verweist, der in den **itemData**-Elementen der der SyntaxhervorhebungsXML-Datei definiert ist. Der jeweilige Wert ist ein Unterobjekt mit den Schlüsseln *text-color*, *selected-text-color*, *bold*, *italic*, *underline*, *strike-through*, *background-color* und *selected-background-color*, definiert im vorherigen Abschnitt. Jeder dieser Werte ist optional, denn wenn sie nicht vorhanden sind, wird der in **text-styles** eingestellte Stil berücksichtigt.

Zum Beispiel hat die Syntaxhervorhebungsdefinition für „ISO C++" in diesem Quelltext einen speziellen Textstil für die Attribute „Type Modifiers" und „Standard Classes". In der zugehörigen XML-Datei „isocpp.xml" verwendet das definierte Attribut „Standard Classes" den Standardstil **BuiltIn** (oder dsBuiltIn). In diesem Attribut wird nur der Wert von **text-color** durch die neue Farbe „#6431b3" überschrieben .

```
"custom-styles": {
    "ISO C++": {
        "Standard Classes": {
            "text-color": "#6431b3"
        },
        "Type Modifiers": {
            "bold": true,
            "selected-text-color": "#009183",
            "text-color": "#00b5cf"
        }
    }
}
```

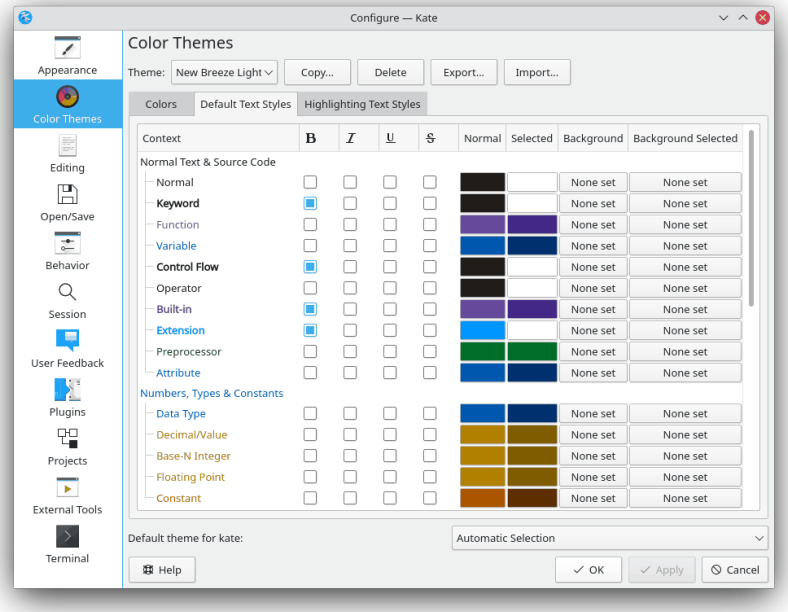
ANMERKUNG

- Sie sollten beachten, dass diese Textstile den Attributnamen zugeordnet sind, die in den XML - Syntaxhervorhebungsdateien definiert sind. Wenn eine XML-Datei aktualisiert und einige Attribute umbenannt oder entfernt werden, ist der im eigenen Schema definierte Stil nicht mehr gültig.
- Definitionen für Syntaxhervorhebung enthalten oft andere Definitionen. Die Syntaxhervorhebung für „QML" zum Beispiel enthält die Syntaxhervorhebung für „JavaScript", da beide die gleichen Funktionen für die Hervorhebung verwenden.

In der GUI zum Verwalten von Schemata von KTextEditor können Sie diese Attribute auf der Karteikarte **`Textstile für Hervorhebungen`** verwalten. Als Voreinstellung wird die Hervorhebung des aktuellen Dokuments vom Editor eingestellt. Sie werden feststellen, dass viele Hervorhebungen andere Hervorhebungen enthalten, die durch Gruppen in der Stilliste repräsentiert werden. Zum Beispiel importieren die meisten Hervorhebungen die Hervorhebung „Alert" und viele Quelltextformate importieren die Hervorhebung „Doxygen".

6.3.5 Die GUI der Farbschemata

Farbschemata lassen sich am einfachsten über die GUI im „Einrichtungs"-Dialog von KTextEditorerstellen und bearbeiten. .Um diesen Dialog zu öffnen, wählen Sie **Einstellungen** → ***Anwendung*** **einrichten ...** aus der Menüleiste in Ihrer Textverarbeitung. Wählen Sie dann **Farbschema** in der Seitenleiste.



Kates Einrichtungsdialog mit der Bearbeitung von Farbschemata.

In diesem Dialog können Sie alle alle Farben in einem beliebigen Schema einstellen, neue Schemata erstellen/kopieren, löschen, sie in eine **`.theme`**-Datei im JSON Format exportieren oder aus externen **`.theme`**-Dateien importieren. Jedes Schema hat Einstellungen für Textfarben und Stile.

Die eingebauten Schemata können als Voreinstellung nicht geändert werden. Dazu müssen Sie sie unter einem neuen Namen kopieren.

Um ein Farbschema dauerhaft in im Texteditor zu verwenden, müssen Sie es im Kombinationsfeld mit der Bezeichnung **Standardschema für *Anwendung*** am unteren Rand des Dialogs auswählen und dann **Anwenden** oder **OK** drücken. Standardmäßig wird die **Automatische Auswahl** aktiviert. Damit wird ein geeigneteres Farbschema passend zum *KDE Plasma-Farbschema* für die Textbearbeitung gewählt. Normalerweise wird zwischen „Breeze-Hellt" und „Breeze-Dunkel" gewählt.

> TIP
> Sie können das globale Farbschema für KDE im Modul Farben in den Systemeinstellungen anpassen. Für einzelne Anwendungen wie Kate und KDevelop können Sie ein Schema auch mit **Einstellungen** → **Farbschema** ändern.

6.3.5.1 Ein neues Schema erstellen

Um ein neues Farbschema zu erstellen, müssen Sie zuerst ein vorhandenes Schema kopieren, Wählen Sie ein vorhandenes Schema aus, das Sie als Grundlage verwenden möchten wie „Breeze-Hell" oder „Breeze-Dunkel". Klicken Sie dann auf **Kopieren** und geben den Namen für das neue Schema ein.

Möchten Sie ein eingebautes oder nur lesbares Schema verändern, müssen Sie es zuerst unter einem anderen Namen kopieren.

6.3.5.2 JSON-Schemadateien importieren oder exportieren

Mit **Exportieren** können Sie ein ausgewähltes Schema einschließlich der eingebauten Schemata in eine JSON-Datei mit der Dateierweiterung **.theme** exportieren. Damit öffnen Sie einen Dialog zum Speichern der Datei. Um eine Farbschema aus einer externen JSON-Datei hinzuzufügen, drücken Sie den Knopf **Importieren** und wählen dann die **.theme**-Datei im Dialog.

> TIP
>
> - Wie oben erwähnt, werden benutzerdefinierte Farbschema-Dateien im Ordner `org.kde.syntax-highlighting/themes/` abgelegt. Wenn Sie ein Farbschema kopieren oder neu erstellen, wird es automatisch dort gespeichert. Auch das Importieren oder Hinzufügen eines Farbschemas entspricht dem Kopieren einer externen **.theme**-Datei in diesen Ordner. KSyntaxHighlighting holt sich automatisch Schemadateien aus diesem Ordner.
> - Wenn Sie ein von Ihnen erstelltes Schema veröffentlichen wollen, ist es unbedingt erforderlich, das Metadaten-Objekt der JSON Datei zu prüfen, eine passende Lizenz hinzuzufügen und die Revisionsnummer zu überprüfen.

6.3.5.3 Bearbeitung von Farbschemata

6.3.5.3.1 Farben

Hier können die Farben des Editorbereichs angepasst werden. Mehr Informationen dazu finden Sie in Abschnitt 6.3.4.1.

6.3.5.3.2 Standardtextstile

Die Stile für Standardtext sind von den Stilen für Hervorhebungen abgeleitet, sodass der Editor Texte immer in der gleichen Form anzeigen kann. So sind zum Beispiel Kommentare unabhängig

vom Textformat oder der Programmiersprache des Quelltextdokuments immer in der gleichen Farbe gekennzeichnet.

Der Name in der Liste der Stile wird so wie Elemente im Dokument mit diesem Kontext angezeigt. So erhalten Sie sofort eine Vorschau beim Bearbeiten.

Zu jedem Stil können Sie Eigenschaften sowie Vordergrund- und Hintergrundfarbe einstellen. Um eine Hintergrundfarbe zu löschen, benutzen Sie die rechte Maustaste, um das Kontextmenü aufzurufen.

Die Attribute in diesem Bereich werden in Abschnitt 6.3.4.2 erläutert.

6.3.5.3.3 Textstile für Hervorhebungen

Hier können Sie die Textstile für bestimmte Hervorhebungsdefinitionen einstellen. Der Editor startet diese Seite mit der Hervorhebung für das aktuelle Dokument. Wenn Sie an einer anderen Hervorhebungsdefinition Veränderungen vornehmen wollen, dann wählen Sie diese mit dem Auswahlfeld **Hervorhebung** aus.

Der Name in der Liste der Stile wird so wie Elemente im Dokument mit diesem Kontext angezeigt. So erhalten Sie sofort eine Vorschau beim Bearbeiten.

Zu jedem Stil können Sie Eigenschaften sowie Vordergrund- und Hintergrundfarbe einstellen. Um eine Hintergrundfarbe zu löschen, benutzen Sie die rechte Maustaste, um das Kontextmenü aufzurufen. Zusätzlich gibt es noch ein Feld, das anzeigt, ob der eingestellte Stil der Standarddefinition entspricht - wenn nicht klicken Sie einfach auf dieses Feld, um die Standardeinstellungen herzustellen.

Sie werden feststellen, dass viele Hervorhebungen andere Hervorhebungen enthalten, die in Untergruppen geordnet sind. So werden zum Beispiel die Hervorhebungen für Warnungen (Alerts) in die meisten Hervorhebungen importiert, viele Quelltexte importieren außerdem die Hervorhebungen für Doxygen. Wenn Sie Änderungen an den importierten Hervorhebungen vornehmen, dann werden nur die Stile im bearbeiteten Format beeinflusst. Andere Formate, die die gleichen Hervorhebungen importiert haben, werden nicht beeinflusst.

6.3.6 Tipps & Tricks

6.3.6.1 Kontrast von Textfarben

Ein wichtiger Aspekt bei der Arbeit mit Farbschemata ist die Wahl eines Textkontrasts, der die Lesbarkeit des Textes erleichtert, insbesondere in Kombination mit dem Hintergrund.

`Kontrast` ist eine Anwendung zur Überprüfung von Farbkontrasten. Damit erkennen Sie, ob die Kombinationen aus Textfarbe und Hintergrundfarbe lesbar und zugänglich sind, daher ist dies ein ausgezeichnetes Werkzeug für die Erstellung von Farbschemata.

Sie können **`Kontrast`** von der KDE-Webseite mit Anwendungen oder als Flatpak-Paket auf Flathub herunterladen (nur für GNU/Linux).

Die GNOME-Anwendung **`Contrast`** funktioniert ähnlich. Sie können sie als Flatpak-Paket auf Flathub herunterladen (nur für GNU/Linux).

6.3.6.2 Vorschläge zur Konsistenz bei der Syntaxhervorhebung

KSyntaxHighlighting enthält mehr als 300 Definitionen für Syntaxhervorhebung, daher sollten Sie dafür sorgen, dass Ihr neues Schema in allen Definitionen der Syntaxhervorhebung gut aussieht. Die eingebauten Farbschemata haben folgende Gemeinsamkeiten, die Sie beachten sollten, um eine eine korrekte Darstellung aller Definitionen der Syntaxhervorhebung zu erhalten:

- Verwenden Sie Fettschrift für „Keyword" und „ControlFlow" als Textstile.

- Benutzen Sie keine Hintergrundfarbe in einem Textstil, außer bei „Alert" und „RegionMarker".

Die meisten der Syntaxhervorhebungen sind für die Standarddesigns „Breeze-Hell" und „Breeze-Dunkel" optimiert.Daher sollten Sie für die Einheitlichkeit ähnlicher Farben verwenden wie in den Textstilen, wie *grün* für „Präprozessor" und „Andere", *blau* für „Datentyp" und „Attribut" oder *purpur* für „Funktion".

Beachten Sie, dass diese Empfehlungen nicht zwingend für das Erstellen und Veröffentlichen eines Schemas sind.

6.4 Scripting mit JavaScript

Die Editorkomponente von KatePart kann durch Skripte erweitert werden. Als Skriptsprache wird ECMAScript verwendet, das auch unter dem Namen JavaScript bekannt ist. In KatePart können zwei Arten von Skripten benutzt werden: Einrückungs- und Befehlszeilenskripte.

6.4.1 Einrückungsskripte

Mit Einrückungsskripten wird der Quelltext automatisch bei der Eingabe eingerückt. Nach Drücken der Eingabetaste wird die Einrückungstiefe zum Beispiel oft vergrößert.

Die folgenden Abschnitte beschreiben Schritt für Schritt, wie das Gerüst für ein einfaches Einrückungsskript entsteht. Zuerst wird eine neue `*.js`-Datei z. B. mit dem Namen `$XDG_DATA_HOME /katepart5/script/indentation` erzeugt. Darin wird die Umgebungsvariable `XDG_DATA_HOME` normalerweise entweder zu `~/.local` oder `~/.local/share` erweitert.

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\katepart5\indentation`. Dabei ist `%USER%` normalerweise `C:\\Users\\user`.

6.4.1.1 Der Vorspann des Einrückungsskripts

Der Vorspann der Datei `javascript.js` ist als JSON am Anfang des Dokumentes eingebettet und hat folgende Form:

```
var katescript = {
    "name": "JavaScript",
    "author": "Example Name <example.name@some.address.org>",
    "license": "BSD License",
    "revision": 1,
    "kate-version": "5.1",
    "required-syntax-style": "javascript",
    "indent-languages": ["javascript"],
    "priority": 0,
}; // kate-script-header, muss am Anfang der Datei stehen und darf keine ↩
    Kommentare enthalten
```

Jede Zeile wird jetzt im Detail erklärt:

- `name` [erforderlich]: Dies ist der Name des Skripts, der im Menü **Extras** → **Einrückung** und im Einrichtungsdialog angezeigt wird.
- `author` [optional]: Der Name des Autors und weitere Kontaktinformationen.
- `license` [optional]: Kurzform der Lizenz, wie zum Beispiel BSD-Lizenz oder LGPLv3.

- `revision` [erforderlich]: Die Version des Skripts, sie sollte bei jeder Änderung des Skripts erhöht werden.
- `kate-version` [required]: Minimal erforderliche KatePart-Version.
- `required-syntax-style` [optional]: Der erforderliche Sytaxstil, der zum angebenen `style` in Syntaxhervorhebungs-Stildateien passt. Dies ist wichtig für Einrückungsskripte, die besondere Informationen über die Hervorhebung im Dokument benötigen. Wenn ein erforderlicher Syntaxstil angegeben ist, ist das Einrückungsskript nur verfügbar, wenn auch die zugehörige Hervorhebung aktiviert ist. Dies verhindert ein „nicht definiertes Verhalten" beim Verwenden einer Einrückung ohne das erwartete Hervorhebungschema. Ein Beispiel dafür ist das Einrückungsskript für Ruby, das diese Option in den Dateien `ruby.js` und `ruby.xml` benutzt.
- `indent-languages` [optional]: JSON-Feld von Syntaxstilen, die das Skript richtig einrücken kann, z. B.: `[˜c++˜, ˜java˜]`.
- `priority` [optional]: Wenn es mehrere Einrückungsskripte für eine bestimmte hervorgehobene Datei gibt, bestimmt die Priorität, welches Skript als Standard verwendet wird.

6.4.1.2 Der Quelltext des Einrückungsskripts

Nachdem der Vorspann beschrieben wurde, erklärt dieser Abschnitt wie das Einrückungsskript selbst funktioniert. Die Basisvorlage für den Quelltext sieht so aus:

```
// benötigt die Bibliothek &#8222;katepart js&#8220; z. B. range.js wenn ↩
    Range benutzt wird
require ("range.js");

triggerCharacters = "{}/:;";
function indent(line, indentWidth, ch)
{
    // wird für jedem Zeilenumbruch (ch == '\n') und alle in der
    // globalen Variable triggerCharacters festgelegten Zeichen aufgerufen. ↩
        Wenn ExtrasAusrichten
    // gewählt wird, ist die Variable ch leer, d. h. ch == ''.
    //
    // siehe auch: Skript-API
    return -2;
}
```

Die Funktion `indent()` hat drei Argumente:

- `line`: die Zeile die eingerückt werden soll
- `indentWidth`: die Einrückungstiefe mit der Anzahl der Leerzeichen
- `ch`: entweder das Zeichen für Zeilenumbruch (`ch == '\n'`), ein in `triggerCharacters` festgelegtes Zeichen oder ein leeres Zeichen, wenn der Benutzer die Aktion **Extras** → **Ausrichten** aufgerufen hat.

Der Rückgabewert der Funktion `indent()` bestimmt, wie die Zeile eingerückt wird. Ist dieser Wert eine Ganze Zahl, wird sie wie folgt interpretiert:

- Rückgabewert `-2`: keine Aktion
- Rückgabewert `-1`: Einrückung beibehalten (sucht nach vorherigen nicht leeren Zeilen)
- Rückgabewert `0`: eine Zahl >= 0 gibt die Anzahl der Leerzeichen zum Einrücken an

Alternativ kann ein Feld mit zwei Elementen zurückgegeben werden:

- `Rückgabewert [ indent, align ];`

In diesem Fall ist das erste Element die Einrückungstiefe wie oben mit derselben Bedeutung spezieller Werte. Das zweite Element ist ein absoluter Wert für die Spalte der „Ausrichtung". Ist dieser Wert größer als der Einrückungswert, legt die Differenz der Werte die Anzahl von Leerzeichen fest, die zum ersten Wert für die gesamte Einrückung hinzuaddiert werden. Anderenfalls wird der zweite Rückgabewert ignoriert. Die Benutzung von Tabulator- und Leerzeichen wird oft als „Mischmodus" bezeichnet.

Betrachten Sie das folgende Beispiel: Angenommen das Tabulatorzeichen wird zur Einrückung verwendet und die Tabulatorweite beträgt 4. Hier steht <tab> für den Tabulator und '.' für ein Leerzeichen:

```
1: <tab><tab>foobar("hello",
2: <tab><tab>......."world");
```

Beim Einrücken der Zeile 2 gibt die Funktion `indent()` [8, 15] zurück. Daher werden zwei Tabulatoren für das Einrücken bis Spalte 8 und noch zusätzlich 7 Leerzeichen hinzugefügt. Dadurch steht der zweite Parameter unter dem ersten und bleibt auch so ausgerichtet, wenn die Datei mit einer anderen Tabulatorweite angezeigt wird.

In der Standardinstallation von KDE wird KatePart mit einigen Einrückungsskripten installiert. Die entsprechenden JavaScript-Quelltexte finden Sie unter `$XDG_DATA_DIRS /katepart5/script/indentation`.

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\katepart5\indentation`. Dabei ist *`%USER%`* normalerweise `C:\\Users\\user`.

Bei der Entwicklung eines Einrückungsskripts muss das Skript wieder neu geladen werden, um testen zu können, ob es richtig funktioniert. Anstatt das ganze Programm neu zu starten, wechseln Sie zur Befehlszeile und geben **reload-scripts** ein.

Wenn Sie nützliche Skripte entwickelt haben, sollten Sie darüber nachdenken, sie zum KatePart-Projekt hinzufügen. Schreiben Sie dazu an die Mailingliste.

6.4.2 Befehlszeilenskripte

Da nicht alle gewünschten Funktionen in KatePart eingebaut werden können, ist es möglich, kleine Hilfsskripte für die schnelle Änderung von Textes mit der eingebauten Befehlszeile auszuführen. Der Befehl **sort** ist zum Beispiel als Skript geschrieben. Dieser Abschnitt erklärt, wie `*.js`-Dateien erstellt werden, um die Fähigkeiten von KatePart mit beliebigen Hilfsskripten zu erweitern.

Befehlszeilenskripte werden im gleichen Ordner wie Einrückungsskripte gespeichert. Zuerst erstellen Sie eine neue `*.js`-Datei namens `myutils.js` im lokalen persönlichen Ordner `$XDG_DATA_HOME /katepart5/script/commands`. Darin wird die Umgebungsvariable `XDG_DATA_HOME` normalerweise entweder zu `~/.local` oder `~/.local/share` erweitert.

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\katepart5\commands`. Dabei ist *`%USERPROFILE%`* normalerweise `C:\\Users\\user`.

6.4.2.1 Der Vorspann des Befehlszeilenskripts

Der Vorspann jedes Befehlszeilen-Skripts ist in der JSON-Datei am Anfang des Skripts so eingebettet:

```
var katescript = {
    "author": "Example Name <example.name@some.address.org>",
    "license": "LGPLv2+",
    "revision": 1,
    "kate-version": "5.1",
    "functions": ["sort", "moveLinesDown"],
    "actions": [
        {   "function": "sort",
            "name": "Sort Selected Text",
            "category": "Editing",
            "interactive": "false"
        },
        {   "function": "moveLinesDown",
            "name": "Move Lines Down",
            "category": "Editing",
            "shortcut": "Ctrl+Shift+Down",
            "interactive": "false"
        }
    ]
}; // kate-script-header, muss am Anfang der Datei stehen und darf keine ↩
    Kommentare enthalten
```

Jede Zeile wird jetzt im Detail erklärt:

- `author` [optional]: Der Name des Autors und weitere Kontaktinformationen.
- `license` [optional]: Kurzform der Lizenz, wie zum Beispiel BSD-Lizenz oder LGPLv2.
- `revision` [erforderlich]: Die Version des Skripts, sie sollte bei jeder Änderung des Skripts erhöht werden.
- `kate-version` [required]: Minimal erforderliche KatePart-Version.
- `functions` [erforderlich]: JSON-Feld der Befehle im Skript.
- `actions` [optional]: JSON-Feld mit JSON-Objekten, das die Aktionen festlegt, die im Anwendungsmenü erscheinen sollen. Weitergehende Informationen finden Sie im Abschnitt Kurzbefehle festlegen.

Da der Inhalt von `functions` ein JSON-Feld ist, kann ein einzelnes Skript eine beliebige Anzahl von Befehlen für die Befehlszeile enthalten. Jede Funktion ist durch die eingebaute Befehlszeile in KatePart verfügbar.

6.4.2.2 Der Quelltext des Skripts

Alle im Vorspann aufgeführten Funktionen müssen im Skript implementiert werden. Im oben gezeigten Skript müssen zum Beispiel die beiden Funktionen **sort** und **moveLinesDown** implementiert werden. Alle Funktionen haben folgende Syntax:

```
// benötigt die Bibliothek &#8222;katepart js&#8220; z. B. range.js wenn ↩
    Range benutzt wird
require ("range.js");
function <name>(arg1, arg2, ...)
{
    // ... Implementierung, siehe auch: Skript-API
}
```

Argumente in der Befehlszeile werden der Funktion als *arg1*, *arg2* usw. übergeben. Um für jeden Befehl eine Dokumentation zu Verfügung zu stellen, verwenden Sie die Funktion „help“ wie im folgenden Beispiel:

```
function help(cmd)
{
    if (cmd == "sort") {
        return i18n("Sortiert den ausgewählten Text.");
    } else if (cmd == "...") {
        // ...
    }
}
```

Durch den Aufruf von **help sort** auf der Befehlszeile wird dann diese Hilfefunktion mit dem Argument *cmd* für den verwendeten Befehl benutzt, d. h. mit *cmd == ˜sort˜*. zeigt Kate den zurückgegebenen Text als Hilfe für den Benutzer an. Denken Sie daran, die Texte zu übersetzen .

Bei der Entwicklung eines Befehlszeilenskripts muss das Skript wieder neu geladen werden, um testen zu können, ob es richtig funktioniert. Anstatt das ganze Programm neu zu starten, wechseln Sie zur Befehlszeile und geben **reload-scripts** ein.

6.4.2.2.1 Kurzbefehle festlegen

Das Skript braucht einen passenden Skript-Header, der die Skripte über das Anwendungsmenü verfügbar macht. Im Beispiel erscheinen die Funktionen `sort` und `moveLinesDown` im Menü. Das wird durch den folgenden Skript-Header erreicht:

```
var katescript = {
    ...
    "actions": [
        {   "function": "sort",
            "name": "Sort Selected Text",
            "icon": "",
            "category": "Editing",
            "interactive": "false"
        },
        {   "function": "moveLinesDown",
            "name": "Move Lines Down",
            "icon": "",
            "category": "Editing",
            "shortcut": "Ctrl+Shift+Down",
            "interactive": "false"
        }
    ]
};
```

Die Felder für eine Aktion lauten wie folgt:

- `function` [erforderlich]: Die Funktion, die im Menü **Extras** → **Skripte** erscheint.
- `name` [erforderlich]: Der Text, der im Menü Skript angezeigt wird.
- `icon` [optional]: Dieses Symbol erscheint neben dem Text im Menü. Alle KDE-Symbolnamen können hier benutzt werden.
- `category` [optional]: Wenn eine Kategorie angegeben ist, dann erscheint das Skript in einem Untermenü.
- `shortcut` [optional]: Der hier angegebene Kurzbefehl ist der Standard. Beispiel: `Ctrl+Alt+T`. Sehen Sie in der Qt-Dokumentation für weitere Einzelheiten nach.

- `interactive` [optional]: Wenn das Skript Benutzereingaben auf der Befehlszeile braucht, dann setzen Sie diesen Parameter auf `true`.

Wenn Sie nützliche Skripte entwickelt haben, sollten Sie darüber nachdenken, sie zum KatePart-Projekt hinzufügen. Schreiben Sie dazu an die Mailingliste.

6.4.3 Skript-API

Die hier vorgestellte Programmierschnittstelle (API) ist in allen Skripten verfügbar, d. h. in Einrückungs- und Befehlszeilenskripten. Die Klassen `Cursor` und `Range` werden durch die Bibliotheksdateien in `$XDG_DATA_DIRS /katepart5/libraries` bereit gestellt. Wenn Sie sie in Ihren Skripten verwenden möchten, was für einige der Funktionen `Document` oder `View` erforderlich ist, fügen Sie bitte die benötigte Bibliothek wie folgt ein:

```
// benötigt die Bibliothek &#8222;katepart js&#8220; z. B. range.js wenn ↩
    Range benutzt wird
require ("range.js");
```

Um die Standard-Skript-API mit eigenen Funktionen und Prototypen zu erweitern, erzeugen Sie eine neue Datei im lokalen Einrichtungsordner für KDE `$XDG_DATA_HOME /katepart5/libraries` und schließen Sie sie in Ihr Skript mit folgendem Befehl ein:

```
require ("myscriptnamehere.js");
```

Auf Windows®-Systemen finden Sie diese Dateien unter `%USERPROFILE%\AppData\Local\katepart5\libraries`. Dabei ist *`%USERPROFILE%`* normalerweise `C:\\Users\\user`.

Um vorhandene Prototypen wie `Cursor` oder `Range` zu erweitern, wird empfohlen, *nicht* die globalen `*.js`-Dateien zu ändern. Ändern Sie stattdessen den `Cursor`-Prototyp in JavaScript, nachdem `cursor.js` in Ihrem Skript mit `require` eingefügt ist.

6.4.3.1 Cursor und Bereiche

Da KatePart ein Texteditor ist, basiert die Skript-API soweit möglich auf Cursor und Bereichen. Ein Cursor ist ein einfaches Tupel `(Zeile, Spalte)` für eine Textposition im Dokument. Ein Bereich umfasst Text vom Start bis zum Ende der Cursor-Position. Die Programmschnittstelle wird in den nächsten Abschnitten im Einzelnen erläutert.

6.4.3.1.1 Der Cursor-Prototyp

`Cursor();`

Konstruktor. Gibt einen Cursor an der Position `(0, 0)` zurück.

Beispiel: `var cursor = new Cursor();`

`Cursor(int zeile, int spalte);`

Konstruktor. Gibt einen Cursor an der Position `(zeile, spalte)` zurück.

Beispiel: `var cursor = new Cursor(3, 42);`

`Cursor(Cursor other);`

Kopierkonstruktor. Gibt die Kopie des Cursors *`other`* zurück.

Beispiel: `var copy = new Cursor(cursor);`

`Cursor Cursor.clone();`

Gibt einen Klon des Cursors zurück.
Beispiel: `var clone = cursor.clone();`

`Cursor.setPosition(int zeile, int spalte);`

Setzt die Cursor-Position auf *`zeile`* und *`spalte`*.
Since: KDE 4.11

`bool Cursor.isValid();`

Überprüft, ob der Cursor gültig ist. Ein Cursor ist ungültig, wenn die Zeile und oder die Spalte den Wert `-1` haben.

Beispiel: `var valid = cursor.isValid();`

`Cursor Cursor.invalid();`

Gibt einen neuen ungültigen Cursor an der Position `(-1, -1)` zurück.

Beispiel: `var invalidCursor = cursor.invalid();`

`int Cursor.compareTo(Cursor other);`

Vergleicht diesen Cursor mit dem Cursor *`other`*. Gibt folgende Werte zurück:

- `-1`, wenn dieser Cursor sich vor dem Cursor *`other`* befindet,
- `0`, wenn beide Cursor an der gleichen Stelle stehen und
- `+1`, wenn dieser Cursor sich hinter dem Cursor *`other`* befindet.

`bool Cursor.equals(Cursor other);`

Gibt `true` zurück, wenn dieser Cursor und der Cursor *`other`* gleich sind, sonst `false`.

`String Cursor.toString();`

Gibt den Cursor als Zeichenfolge in der Form „`Cursor(zeile, spalte)`" zurück.

6.4.3.1.2 Der Bereich-Prototyp

`Range();`

Konstruktor. Der Aufruf `new Range()` gibt einen Bereich von (0, 0) - (0, 0) zurück.

`Range(Cursor start, Cursor ende);`

Konstruktor. Der Aufruf `new Range(start, end)` gibt einen Bereich von (*`start`*, *`ende`*) zurück.

`Range(int startZeile, int startspalte, int endZeile, int endSpalte);`

Konstruktor. Der Aufruf von `new Range(startZeile, startSpalte, endZeile, endSpalte)`. Gibt den Bereich von (*`startZeile`*, *`startSpalte`*) bis (*`endZeile`*, *`endSpalte`*) zurück.

`Range(Range other);`

Kopierkonstruktor. Gibt eine Kopie von Range *`other`* zurück.

`Range Range.clone();`

Gibt einen Klon des Bereichs zurück.
Beispiel: `var clone = range.clone();`

`bool Range.isEmpty();`

Gibt `true` zurück, wenn der Start- und der End-Cursor gleich sind.

Beispiel: `var empty = range.isEmpty();`

Since: KDE 4.11

`bool Range.isValid();`

Gibt `true` zurück, wenn sowohl Start- als auch End-Cursor gültig sind, sonst `false`.

Beispiel: `var valid = range.isValid();`

`Range Range.invalid();`

Gibt den Bereich von (-1, -1) bis (-1, -1) zurück.

`bool Range.contains(Cursor cursor);`

Gibt `true` zurück, wenn dieser Bereich die Cursor-Position enthält, sonst `false`.

`bool Range.contains(Range other);`

Gibt `true` zurück, wenn dieser Bereich den Bereich *`other`* enthält, sonst `false`.

`bool Range.containsColumn(int spalte);`

Gibt `true` zurück, wenn *`spalte`* in dem halboffenen Intervall `[start.spalte, end.spalte)` liegt, sonst `false`.

`bool Range.containsLine(int zeile);`

Gibt `true` zurück, wenn *`zeile`* in dem halboffenen Intervall `[start.zeile, end.zeile)` liegt, sonst `false`.

`bool Range.overlaps(Range other);`

Gibt `true` zurück, wenn dieser Bereich und der Bereich *`other`* sich überlappen, sonst `false`.

`bool Range.overlapsLine(int zeile);`

Gibt `true` zurück, wenn *`zeile`* in dem Intervall `[start.zeile, end.zeile]` liegt, sonst `false`.

`bool Range.overlapsColumn(int spalte);`

Gibt `true` zurück, wenn *`spalte`* in dem Intervall `[start.spalte, end.spalte]` liegt, sonst `false`.

`bool Range.onSingleLine();`

Gibt `true` zurück, wenn der Bereich in der gleichen Zeile beginnt und endet, d. h. wenn *`Range.start.line == Range.end.line`* ist.

Seit: KDE 4.9

`bool Range.equals(Range other);`

Gibt `true` zurück, wenn dieser Bereich und der Bereich *`other`* gleich sind, sonst `false`.

`String Range.toString();`

Gibt den Bereich als Zeichenfolge in der Form „`Range(Cursor(zeile, spalte), Cursor(zeile, spalte))`" zurück.

6.4.3.2 Globale Funktionen

Dieser Abschnitt listet alle globalen Funktionen auf.

6.4.3.2.1 Lesen & Einfügen von Dateien

`String read(String datei);`

Sucht nach der angegebenen *`datei`* relativ zum Ordner `katepart/script/files` und gibt den Inhalt der Datei als String zurück.

`void require(String datei);`

Sucht die angegebene *`datei`* relativ zum Ordner `katepart/script/libraries` und wertet sie aus. `require` verhindert intern, dass die gleiche *`datei`* mehrfach eingeschlossen wird.

Seit: KDE 4.10

6.4.3.2.2 Fehlersuche

`void debug(String text);`

Gibt den *`text`* auf der Standardausgabe `stdout` in der Konsole aus, von der das Programm gestartet wurde.

6.4.3.2.3 Übersetzung

Die Lokalisierung wird durch einige Funktionen zum Übersetzen von Zeichenfolgen in Skripten unterstützt, so durch `i18n`, `i18nc`, `i18np` und `i18ncp`. Diese Funktionen arbeiten genau wie auf der Seite KDE's Übersetzungsfunktionen beschrieben.

Die Übersetzungsfunktionen übersetzen die eingepackten Zeichenfolgen durch KDE's Übersetzungssystem in die Sprache, die in der Anwendung benutzt wird. Zeichenfolgen in Skripten, die in den offiziellen KatePart-Quellen entwickelt werden, werden automatisch extrahiert und sind übersetzbar. Anders gesagt, müssen Sie sich als Entwickler von KatePart nicht um das Extrahieren und Übersetzen kümmern. Achtung, diese Funktionalität gibt es nur innerhalb der KDE-Infrastruktur. Neue Texte in Skripten, die von Anderen ausserhalb von KDE entwickelt wurden, werden nicht automatisch übersetzt. Bitte denken Sie darüber nach, solche Skripte an KDE zu geben, so dass die korrekte Übersetzung möglich wird.

`void i18n(String text, arg1, ...);`

Übersetzt *`text`* in die von der Anwendung benutzte Sprache. Die Argumente *`arg1`*, ..., sind optional und werden benutzt, um die Platzhalter `%1`, `%2` usw. zu ersetzen.

`void i18nc(String context, String text, arg1, ...);`

Übersetzt *`text`* in die von der Anwendung benutzte Sprache. Zusätzlich ist die Zeichenfolge *`context`* sichtbar, so dass Übersetzer bessere Übersetzungen anfertigen können. Die Argumente *`arg1`*, ..., sind optional und werden benutzt, um die Platzhalter `%1`, `%2` usw.. zu ersetzen.

`void i18np(String singular, String plural, int number, arg1, ...);`

Übersetzt entweder *`singular`* oder *`plural`* in die von der Anwendung benutzte Sprache. Dies ist abhängig von der angegebenen *`number`*. Die Argumente *`arg1`*, ..., sind optional und werden benutzt, um die Platzhalter `%1`, `%2` usw.. zu ersetzen.

`void i18ncp(String context, String singular, String plural, int number, arg1, ...);`

Übersetzt entweder *`singular`* oder *`plural`* in die von der Anwendung benutzte Sprache. Dies ist abhängig von der angegebenen *`number`*. Zusätzlich ist die Zeichenfolge *`context`* sichtbar, so dass Übersetzer bessere Übersetzungen anfertigen können. Die Argumente *`arg1`*, ..., sind optional und werden benutzt, um die Platzhalter `%1`, `%2` usw.. zu ersetzen.

6.4.3.3 Die Programmschnittstelle zur Ansicht

Für jedes ausgeführte Skript gibt es eine globale Variable „`view`", für die aktuelle Editoransicht. Im Folgenden finden Sie eine Liste aller verfügbaren Funktionen für eine Ansicht.

`void view.copy()`

Kopiert eine vorhandene Auswahl, ansonsten die aktuelle Zeile, wenn die Einstellung **`[ ] Die aktuelle Zeile kopieren/ausschneiden, wenn keine Markierung vorliegt`** aktiviert ist.

Seit KDE Frameworks 5.79

void view.cut()

Schneidet eine vorhandene Auswahl, ansonsten die aktuelle Zeile aus, wenn die Einstellung **[ ] Die aktuelle Zeile kopieren/ausschneiden, wenn keine Markierung vorliegt** aktiviert ist.

Seit KDE Frameworks 5.79

void view.paste()

Inhalt der Zwischenablage einfügen.

Seit KDE Frameworks 5.79

Cursor view.cursorPosition()

Gibt die aktuelle Position des Cursors in der Ansicht zurück.

void view.setCursorPosition(*int* zeile, *int* spalte); void view.setCursorPosition(*Cursor* cursor);

Setzt die aktuelle Position des Cursors entweder auf (zeile, spalte) oder auf den angegebenen Cursor.

Cursor view.virtualCursorPosition();

Gibt die virtuelle Cursor-Position zurück, dabei wird jeder Tabulator mit der Anzahl der Leerzeichen entsprechend der aktuellen Tabulatorweite berechnet.

void view.setVirtualCursorPosition(*int* zeile, *int* spalte); void view.setVirtualCursorPosition(*Cursor* cursor);

Setzt die aktuelle Position des virtuellen Cursors entweder auf (zeile, spalte) oder auf den angegebenen Cursor.

String view.selectedText();

Gibt den ausgewählten Text zurück. Ist kein Text ausgewählt, wird eine leere Zeichenfolge zurückgegeben.

bool view.hasSelection();

Gibt `true` zurück, wenn die Ansicht ausgewählten Text enthält, sonst `false`.

Range view.selection();

Gibt den ausgewählten Textbereich zurück. Der zurückgegebene Bereich ist ungültig, wenn kein Text ausgewählt ist.

void view.setSelection(*Range* range);

Setzt den ausgewählten Text zum angegebenen Bereich.

void view.removeSelectedText();

Entfernt den ausgewählten Text. Wenn in der Ansicht kein Text ausgewählt ist, passiert nichts.

void view.selectAll();

Wählt den gesamten Text im Dokument aus.

void view.clearSelection();

Löscht die Textauswahl, aber nicht den Text selbst.

object view.executeCommand(*String* command, *String* args, *Range* range);

Führt den Befehl *command* mit den optionalen Argumenten *args* und *range* aus. Das zurückgegebene *object*hat die boolesche Eigenschaft *object.ok*, mit der angegeben wird, ob der Befehl *command* erfolgreich ausgeführt wurde. Bei einem Fehler enthält *object.status* die Fehlermeldung.

Seit KDE Frameworks 5.50

6.4.3.4 Die Programmschnittstelle zum Dokument

Für jedes ausgeführte Skript gibt es eine globale Variable „`document`", die das aktuelle Dokument verweist. Im Folgenden finden Sie eine Liste aller verfügbaren Funktionen für ein Dokument.

`String document.fileName();`

Gibt den Dateinamen des Dokuments zurück oder eine leere Zeichenfolge für nicht gespeicherte Textpuffer.

`String document.url();`

Gibt die vollständige URL des Dokuments zurück oder eine leere Zeichenfolge für nicht gespeicherte Textpuffer.

`String document.mimeType();`

Gibt den MIME-Typ des Dokuments zurück oder `application/octet-stream`, wenn kein passender MIME-Typ gefunden wurde.

`String document.encoding();`

Gibt die aktuell verwendete Kodierung zum Speichern der Datei zurück.

`String document.highlightingMode();`

Gibt den globalen Hervorhebungsmodus für das gesamte Dokument zurück.

`String document.highlightingModeAt(Cursor pos);`

Gibt den Hervorhebungsmodus an der angegebenen Cursor-Position im Dokument zurück.

`Array document.embeddedHighlightingModes();`

Gibt ein Feld von Hervorhebungsmodi zurück, die in diesem Dokument eingebettet sind..

`bool document.isModified();`

Gibt `true` zurück, wenn das Dokument ungespeicherte Änderungen enthält, sonst `false`.

`String document.text();`

Gibt den gesamten Inhalt des Dokuments in einer einzigen Zeichenfolge zurück. Zeilenumbrüche werden mit dem zugehörigen Zeichen „`\n`" markiert.

`String document.text(int vonZeile, int vonSpalte, int bisZeile, int bisSpalte); String document.text(Cursor von, Cursor bis); String document.text(Range range);`

Gibt den Text im angegebenen Bereich zurück. Es wird empfohlen, die Cursor- und Bereichsbasierte Version zu benutzen, dadurch ist der Quelltext besser lesbar.

`String document.line(int zeile);`

Gibt die angegebene Textzeile als Zeichenfolge zurück. Die Zeichenfolge ist leer, wenn die angeforderte Zeile außerhalb des Bereichs liegt.

`String document.wordAt(int zeile, int spalte); String document.wordAt(Cursor cursor);`

Gibt das Wort an der angegebenen Cursor-Position zurück.

`Range document.wordRangeAt(int zeile, int spalte); Range document.wordRangeAt(Cursor cursor);`

Gibt den Bereich des Wortes an der angegebenen Cursor-Position zurück. Der zurückgegebene Bereich ist ungültig (siehe Range.isValid()), wenn die Textposition nach dem Zeilenende liegt. Befindet sich an der angegebenen Cursor-Position kein Wort, wird ein leere Bereich zurückgegeben.

Seit: KDE 4.9

String document.charAt(*int* zeile, *int* spalte); String document.charAt(*Cursor* cursor);

Gibt das Zeichen an der aktuellen Cursor-Position zurück.

String document.firstChar(*int* zeile);

Gibt in der angegebenen *zeile* das erste Zeichen zurück, das kein Leerraumzeichen ist. Wenn die Zeile leer ist oder nur Leerraumzeichen enthält, wird eine leere Zeichenfolge zurückgegeben.

String document.lastChar(*int* zeile);

Gibt in der angegebenen *zeile* das letzten Zeichen zurück, das kein Leerraumzeichen ist. Wenn die Zeile leer ist oder nur Leerraumzeichen enthält, wird eine leere Zeichenfolge zurückgegeben.

bool document.isSpace(*int* zeile, *int* spalte); bool document.isSpace(*Cursor* cursor);

Gibt `true` zurück, wenn das Zeichen an der angegebenen Cursor-Position ein Leerraumzeichen ist, sonst `false`.

bool document.matchesAt(*int* line, *int* column, *String* text); bool document.matchesAt(*Cursor* cursor, *String* text);

Gibt `true` zurück, wenn der angegebene *text* mit der zugehörigen Cursor-Position übereinstimmt, sonst `false`.

bool document.startsWith(*int* zeile, *String* text, *bool* skipWhiteSpaces);

Gibt `true` zurück, wenn die Zeile mit *text* beginnt, sonst `false`. Das Argument *skipWhiteSpaces* bestimmt, ob führende Leerraumzeichen ignoriert werden.

bool document.endsWith(*int* zeile, *String* text, *bool* skipWhiteSpaces);

Gibt `true` zurück, wenn die Zeile mit *text* endet, sonst `false`. Das Argument *skipWhiteSpaces* bestimmt, ob angehängte Leerraumzeichen ignoriert werden.

bool document.setText(*String* text);

Setzt den Text für das gesamte Dokument.

bool document.clear();

Löscht den gesamten Text im Dokument.

bool document.truncate(*int* zeile, *int* spalte); bool document.truncate(*Cursor* cursor);

Schneidet die angegebene Zeile an der Spalte oder an der Cursor-Position ab. War das erfolgreich, wird `true` zurückgegeben, oder `false`, wenn die angegeben Zeile nicht im Bereich des Dokuments liegt.

bool document.insertText(*int* zeile, *int* spalte, *String* text); bool document.insertText(*Cursor* cursor, *String* text);

Fügt den *text* an der angegebenen Cursor-Position ein. War das erfolgreich, wird `true` zurückgegeben, oder `false`, wenn das Dokument im Nur-Lesen-Modus geöffnet wurde.

bool document.removeText(*int* vonZeile, *int* vonSpalte, *int* bisZeile, *int* bisSpalte); bool document.removeText(*Cursor* von, *Cursor* bis); bool document.removeText(*Range* bereich);

Löscht den Text im angegebenen Bereich. War das erfolgreich, wird `true` zurückgegeben, oder `false`, wenn das Dokument im Nur-Lesen-Modus geöffnet wurde.

bool document.insertLine(*int* zeile, *String* text);

Fügt Text in einer angegebenen Zeile ein. War das erfolgreich, wird `true` zurückgegeben, oder `false`, wenn das Dokument im Nur-Lesen-Modus geöffnet wurde oder die Zeile nicht mehr im Bereich des Dokuments liegt.

bool document.removeLine(*int* zeile);

Löscht die angegebene Textzeile. War das erfolgreich, wird `true` zurückgegeben, oder `false`, wenn das Dokument im Nur-Lesen-Modus geöffnet wurde oder die Zeile nicht mehr im Bereich des Dokuments liegt.

bool document.wrapLine(*int* zeile, *int* spalte); bool document.wrapLine(*Cursor* cursor);

Bricht die Zeile an der angegebenen Cursor-Position um. War das erfolgreich, wird `true` zurückgegeben, ansonsten `false`, z. B. wenn die angegeben Zeilennummer < 0 ist.

Seit: KDE 4.9

void document.joinLines(*int* startZeile, *int* endZeile);

Verbindet die Zeilen von *`startZeile`* bis *`endZeile`*. Zwei aufeinanderfolgende Textzeilen werden immer mit einem einzelnen Leerzeichen getrennt.

int document.lines();

Gibt die Zeilenanzahl des Dokuments zurück.

bool document.isLineModified(*int* zeile);

Gibt `true` zurück, wenn die *`zeile`* noch nicht gespeicherte Daten enthält.

Seit: KDE 5.0

bool document.isLineSaved(*int* zeile);

Gibt `true` zurück, wenn *`zeile`* geändert und das Dokument gespeichert wurde. Folglich enthält die aktuelle Zeile keine ungesicherten Daten.

Seit: KDE 5.0

bool document.isLineTouched(*int* zeile);

Gibt `true` zurück, wenn *`zeile`* ungesicherte Daten enthält oder geändert wurde.

Seit: KDE 5.0

void document.findTouchedLine(*int* startZeile, *bool* nach unten);

Suche nach der nächsten Zeile, die ungesicherte Daten enthält oder geändert wurde. Die Suche wird in der *`startZeile`* begonnen und in der Richtung durchgeführt, die in *`nach unten`* angegeben ist.

Seit: KDE 5.0

int document.length();

Gibt die Anzahl der Zeichen des Dokuments zurück.

int document.lineLength(*int* zeile);

Gibt die Länge der *`zeile`* zurück.

void document.editBegin();

Beginnt eine Bearbeitungsgruppe für die Gruppierung von Rückgängig/Wiederherstellen. Achten Sie darauf, `editEnd()` immer genauso oft wie `editBegin()` zu benutzen. Der Aufruf von `editBegin()` verwendet intern einen Referenzzähler, d. h. diese Aufrufe können geschachtelt werden.

void document.editEnd();

Beendet eine Bearbeitungsgruppe. Der letzte Aufruf von `editEnd()` (d. h. der Aufruf zum ersten Aufruf von `editBegin()`) beendet den Bearbeitungsschritt.

int document.firstColumn(*int* zeile);

Gibt die erste Spalte in der angegebenen *`zeile`* zurück, die kein Leerraumzeichen enthält. Besteht die Zeile nur aus Leerraumzeichen, wird `-1` zurückgegeben.

int document.lastColumn(*int* zeile);

Gibt die letzte Spalte in der angegebenen *`zeile`* zurück, die kein Leerraumzeichen enthält. Besteht die Zeile nur aus Leerraumzeichen, wird `-1` zurückgegeben.

int document.prevNonSpaceColumn(*int* zeile, *int* spalte); int document.prevNonSpaceColumn(*Cursor* cursor);

Gibt die Spalte zurück, die keine Leerraumzeichen enthält. Die Suche beginnt an der angegebenen Cursor-Position und erfolgt dabei rückwärts.

int document.nextNonSpaceColumn(*int* zeile, *int* spalte); int document.nextNonSpaceColumn(*Cursor* cursor);

Gibt die Spalte zurück, die keine Leerraumzeichen enthält. Die Suche beginnt an der angegebenen Cursor-Position und erfolgt dabei vorwärts.

int document.prevNonEmptyLine(*int* zeile);

Gibt die nächste nicht leere Zeile zurück, die keine Leerraumzeichen enthält. Die Suche erfolgt dabei rückwärts.

int document.nextNonEmptyLine(*int* zeile);

Gibt die nächste nicht leere Zeile zurück, die keine Leerraumzeichen enthält. Die Suche erfolgt dabei vorwärts.

bool document.isInWord(*String* zeichen, *int* attribut);

Gibt `true` zurück, wenn das angegebene *zeichen* mit den angegebenen *attribut* Teil eines Wortes sein kann, sonst `false`.

bool document.canBreakAt(*String* zeichen, *int* attribut);

Gibt `true` zurück, wenn die Zeile an dem angegebenen *zeichen* mit den angegebenen *attribut* umgebrochen werden kann, sonst `false`.

bool document.canComment(*int* startAttribut, *int* endAttribut);

Gibt `true` zurück, wenn ein mit dem angegebenen Attribut beginnender und endender Bereich auskommentiert werden kann, sonst `false`.

String document.commentMarker(*int* attribut);

Gibt das Kommentarzeichen für einzeilige Kommentare für ein angegebenes *attribut* zurück.

String document.commentStart(*int* attribut);

Gibt das Kommentarzeichen für den Beginn von mehrzeiligen Kommentaren für ein angegebenes *attribut* zurück.

String document.commentEnd(*int* attribut);

Gibt das Kommentarzeichen für das Ende von mehrzeiligen Kommentaren für ein angegebenes *attribut* zurück.

Range document.documentRange();

Gibt einen Bereich zurück, der dass gesamte Dokument umfasst.

Cursor documentEnd();

Gibt einen Cursor zurück, der an der letzten Spalte in der letzten Zeile des Dokuments positioniert ist.

bool isValidTextPosition(*int* zeile, *int* spalte); bool isValidTextPosition(*Cursor* cursor);

Gibt `true` zurück, wenn der Cursor an eine gültigen Position innerhalb eines Textes positioniert ist. Eine Textposition ist nur dann gültig, wenn der Cursor am Anfang, in der Mitte oder am Ende einer gültigen Zeile positioniert ist. Weiterhin ist eine Textposition ungültig, wenn diese in einem Unicode-Surrogat liegt.

Seit: KDE 5.0

int document.attribute(*int* zeile, *int* spalte); int document.attribute(*Cursor* cursor);

Gibt das Attribut an der aktuellen Cursor-Position zurück.

bool document.isAttribute(*int* zeile, *int* spalte, *int* attribut); bool document.isAttribute(*Cursor* cursor, *int* attribut);

Gibt `true` zurück, wenn das Attribut an der angegebenen Cursor-Position gleich *`attribut`* ist, sonst `false`.

String document.attributeName(*int* zeile, *int* spalte); String document.attributeName(*Cursor* cursor);

Gibt den Attributnamen als lesbaren Text zurück. Dies entspricht dem Namen `itemData` in den Syntaxhervorhebungs-Dateien.

bool document.isAttributeName(*int* zeile, *int* spalte, *String* name); bool document.isAttributeName(*Cursor* cursor, *String* name);

Gibt `true` zurück, wenn der Attributname an der angegebenen Cursor-Position gleich *`name`* ist, sonst `false`.

String document.variable(*String* key);

Gibt den Wert der angefragten Dokumentvariablen *`key`* zurück. Existiert diese Variable nicht, wird eine leere Zeichenfolge zurückgegeben.

void document.setVariable(*String* key, *String* value);

Setzt den Wert der angefragten Dokumentvariablen *`key`*.

Siehe auch: Kate-Dokumentvariable
Seit: KDE 4.8

int document.firstVirtualColumn(*int* zeile);

Gibt in der angegebenen Zeile die virtuelle Spalte des ersten Zeichens zurück, das kein Leerraumzeichen ist, oder `-1`, wenn die Zeile leer ist oder nur Leerraumzeichen enthält.

int document.lastVirtualColumn(*int* zeile);

Gibt in der angegebenen Zeile die virtuelle Spalte des letzten Zeichens zurück, das kein Leerraumzeichen ist, oder `-1`, wenn die Zeile leer ist oder nur Leerraumzeichen enthält.

int document.toVirtualColumn(*int* zeile, *int* spalte); int document.toVirtualColumn(*Cursor* cursor); Cursor document.toVirtualCursor(*Cursor* cursor);

Wandelt die angegebene „reale" Cursor-Position in eine virtuelle Cursor-Position um und gibt entweder einen „int"-Wert oder ein Cursor-Objekt zurück.

int document.fromVirtualColumn(*int* zeile, *int* virtuelleSpalte); int document.fromVirtualColumn(*Cursor* virtuellerCursor); Cursor document.fromVirtualCursor(*Cursor* virtuellerCursor);

Wandelt die angegebene virtuelle Cursor-Position in eine „reale" Cursor-Position um und gibt entweder einen „int"-Wert oder ein Cursor-Objekt zurück.

Cursor document.anchor(*int* zeile, *int* spalte, *Char* zeichen); Cursor document.anchor(*Cursor* cursor, *Char* zeichen);

Sucht rückwärts nach dem angegebenen Zeichen und beginnt dabei an dem angegebenen Cursor. Wenn zum Beispiel „(" als Zeichen ist, gibt diese Funktion die Position der öffnenden Klammer „(". Dabei wird das Vorkommen mitgezählt, d. h. andere Klammern „(...)" werden ignoriert.

Cursor document.rfind(*int* zeile, *int* spalte, *String* text, *int* attribut = *-1*); Cursor document.rfind(*Cursor* cursor, *String* text, *int* attribut = *-1*);

Sucht rückwärts nach dem angegeben Text mit dem passenden *`attribut`*. Ein Attribut mit dem Wert `-1`wird dabei ignoriert. Es wird ein ungültiger Cursor zurückgegeben, wenn der Text nicht gefunden wurde.

int document.defStyleNum(*int* zeile, *int* spalte); int document.defStyleNum(*Cursor* cursor);

Gibt den Standardstil zurück, der an der angegebenen Cursor-Position benutzt wird.

bool document.isCode(*int* zeile, *int* spalte); bool document.isCode(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut an der angegeben Cursor-Position nicht den folgenden Stilen entspricht: `dsComment`, `dsString`, `dsRegionMarker`, `dsChar`, `dsOthers`.

bool document.isComment(*int* zeile, *int* spalte); bool document.isComment(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut des Zeichens an der Cursor-Position `dsComment` ist, sonst `false`.

bool document.isString(*int* zeile, *int* spalte); bool document.isString(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut des Zeichens an der Cursor-Position `dsString` ist, sonst `false`.

bool document.isRegionMarker(*int* zeile, *int* spalte); bool document.isRegionMarker(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut des Zeichens an der Cursor-Position `dsRegionMarker` ist, sonst `false`.

bool document.isChar(*int* zeile, *int* spalte); bool document.isChar(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut des Zeichens an der Cursor-Position `dsChar` ist, sonst `false`.

bool document.isOthers(*int* zeile, *int* spalte); bool document.isOthers(*Cursor* cursor);

Gibt `true` zurück, wenn das Attribut des Zeichens an der Cursor-Position `dsOthers` ist, sonst `false`.

6.4.3.5 Die Programmschnittstelle zum Editor

Zusätzlich zur Programmierschnittstelle für das Dokument und die Ansicht gibt es eine weitere Schnittstelle mit Funktionen für Skripte des Editors.

String editor.clipboardText();

Gibt den aktuellen Text aus der globalen Zwischenablage zurück.

Seit KDE Frameworks 5.50

String editor.clipboardHistory();

Der Editor enthält den Verlauf der Zwischenablage mit bis zu 10 Eintragen. Diese Funktion gibt alle Einträge zurück, die aktuell im Verlauf der Zwischenablage vorhanden sind.

Seit KDE Frameworks 5.50

void editor.setClipboardText(*String* text);

Setzt den Inhalt der Zwischenablagen auf den Wert *text*, der auch zum Verlauf hinzugefügt wird.

Seit KDE Frameworks 5.50

Kapitel 7

Einrichten von KatePart

Die Auswahl von **Einstellungen** → ***Anwendung*** **einrichten ...** im Menü öffnet das Einrichtungsfenster;. In diesem Dialogfenster können eine ganze Reihe von Einstellungen vorgenommen werden. Die angezeigten Einstellungsmöglichkeiten hängen von der Auswahl eines links in der Liste angezeigten Symbols ab. Die drei Knöpfe am unteren Rand des Dialogfeldes rufen die **Hilfe** auf, machen die aktuellen Einstellungen mit **OK** gütig, oder brechen mit **Abbrechen** den Einstellungs-Prozess ab.

Sie können das **Hilfe**system aufrufen, die aktuellen Einstellungen mit **OK** übernehmen und das Dialogfeld schließen, oder den Knopf **Abbrechen** benutzen, um das Dialogfeld zu schließen, ohne Änderungen zu speichern. Die zur Auswahl stehenden Kategorien - **Erscheinungsbild**, **Schriften & Farben**, **Bearbeitung**, **Öffnen/Speichern** und **Erweiterungen** sind nachfolgend erläutert.

7.1 Einstellungen für die Editor-Komponente

Diese Gruppe enthält alle Seiten, auf denen die Einstellungen zum Editor von KatePart vorgenommen werden. Für die meisten der Einstellungen gibt es Standardwerte, die durch Festlegen von Datentypen, Dokumentvariablen oder durch dokumentbezogene Einstellungen verändert werden können.

7.1.1 Erscheinungsbild

7.1.1.1 Schriftart

Hier stellen Sie die Schriftarten für den Text im Editor ein. Sie können jede Schriftart verwenden, die auf Ihrem System verfügbar ist und Sie können eine Standardgröße einstellen. Unten im Dialog wird ein Beispiel in der gewählten Schrift angezeigt, sodass Sie die Auswirkungen Ihrer Wahl sofort sehen.
Weitere Informationen dazu finden Sie im Abschnitt Auswahl von Schriftarten der KDE-Grundlagen.

7.1.1.2 Allgemein

Dynamischer Zeilenumbruch

Wenn eingeschaltet, dann werden die Zeilen am rechten Bildschirmrand automatisch umgebrochen.

Kennzeichnung für dynamischen Zeilenumbruch:

Wählen Sie hier, ob die Markierungen für den dynamischen Zeilenumbruch angezeigt werden sollen, entweder **Aus**, **Zeilennummern folgen** or **Immer aktiv**.

Dynamisch umbrochene Zeilen an der Einrückungstiefe ausrichten:

Dynamisch umgebrochene Zeilen werden auf die Einrückungsposition der ersten Zeile des Abschnittes eingerückt. Dadurch werden Quelltexte besser lesbar.

Zusätzlich können sie hier ein Maximum angeben, ab dem die neuen Zeilen nicht weiter eingerückt werden. Wenn Sie hier zum Beispiel 50 % angeben, dann werden Zeilen nicht weiter eingerückt, deren Einrückung weiter als 50 % der Bildschirmbreite sein würde.

Leerraum-Hervorhebung

Tabulatoren hervorheben
Im Editor wird ein »-Symbol für einen vorhandenen Tabulator angezeigt.

Leerzeichen am Zeilenende hervorheben
Im Editor werden Punkte angezeigt, wenn zusätzliche Leerzeichen am Zeilenende vorhanden sind.

Größe der Hervorhebungsmarkierungen

Ändern Sie mit dem Schieberegler die Größe der sichtbaren Markierung für Hervorhebungen.

Erweitert

Einrückungslinien anzeigen

Wenn dieses Feld angekreuzt ist, dann werden im aktuellen Dokument senkrechte Linien angezeigt, die Ihnen helfen, eingerückte Zeilen zuzuordnen.

Bereich zwischen zusammengehörenden Klammern hervorheben

Ist diese Einstellung markiert, wird der Bereich zwischen den ausgewählten, zusammengehörenden Klammern hervorgehoben.

Zusammengehörige Klammern animieren

Ist dies aktiviert, werden mit dem Mauszeiger auf Klammern ({, [,], },(oder)) die zugehörigen schließenden Klammern hervorgehoben.

Erste Zeile ausblenden

Ist die aktiviert, dann wird die erste Zeile ausgeblendet. Benutzen Sie diese Einstellung, wenn die Datei mit einem Kommentar with zum Beispiel einem Copyright beginnt.

Wortanzahl anzeigen

Zeigt die Zahl der Wörter und Zeichen im Dokument und in der aktuellen Auswahl in der Statusleiste. Diese Einstellung finden Sie auch im Kontextmenü der Statusleiste.

Zeilenanzahl anzeigen

Zeigt die Zeilenanzahl im Dokument in der Statusleiste. Diese Einstellung finden Sie auch im Kontextmenü der Statusleiste.

7.1.1.3 Randbereiche

Randbereiche

Markierungen für Quelltextausblendungen anzeigen

Wenn dieses Feld angekreuzt ist, dann werden im aktuellen Dokument für Quelltextausblendungen Markierungen angezeigt.

Vorschau des ausgeblendeten Texts anzeigen

Ist dies aktiviert, wird beim Überfahren eines ausgeblendeten Bereich eine Vorschau des ausgeblendeten Texts in einen Fenster angezeigt.

Symbolrand anzeigen

Wenn dieses Feld angekreuzt ist, dann wird im aktuellen Dokument an der linken Seite der Symbolrand angezeigt. Darin werden zum Beispiel Markierungen für Lesezeichen angezeigt.

Zeilennummern anzeigen

Wenn dieses Feld angekreuzt ist, dann werden im aktuellen Dokument an der linken Seite Zeilennummern angezeigt.

Markierungen für geänderte Zeilen anzeigen

Ist diese Einstellung aktiv, werden Markierungen für geänderte Zeilen angezeigt. Weitere Informationen finden Sie unter Abschnitt 3.9.

Markierung für Bildlaufleiste anzeigen

Wenn dieses Feld angekreuzt ist, dann werden im aktuellen Dokument Markierungen in der senkrechten Bildlaufleiste angezeigt. Diese zeigen zum Beispiel Lesezeichen.

Textvorschau an der Bildlaufleiste anzeigen

Ist diese Einstellung aktiviert, wird beim Überfahren der Bildlaufleiste mit dem Mauszeiger eine verkleinerte Textvorschau mit mehreren Textzeilen um die Position des Mauszeigers angezeigt. Damit können Sie schnell zu anderen Bereichen des Dokuments wechseln.

Textgrafik auf Bildlaufleiste anzeigen

Ist diese Einstellung aktiv, zeigen neu geöffnete Ansichten ein verkleinerte Grafik des Texts im Dokuments auf der senkrechten Bildlaufleiste.
Weitere Informationen über die Textgrafik auf der Bildlaufleiste finden Sie im Abschnitt Abschnitt 3.10.

Breite der Textgrafik:

Bestimmt die Breite der Textgrafik auf der Bildlaufleiste in Pixeln.

Anzeige der Bildlaufleisten

Schaltet die Bildlaufleisten ein, aus oder nur ein, wenn erforderlich. Klicken Sie mit der linken Maustaste auf das blaue Rechteck, dann wird der Zeilenbereich des Dokumentausschnitts auf dem Bildschirm angezeigt. Halten Sie die linke Maustaste außerhalb des blauen Rechtecks, um automatisch durch des Dokument zu blättern.

Lesezeichenmenü sortieren

Nach Erstellungszeitpunkt

Jedes neue Lesezeichen wird am Ende der Liste hinzugefügt.

Nach Position

Die Lesezeichen werden nach Zeilennummern geordnet.

7.1.2 Farbschemata

Dieser Abschnitt erlaubt die Einstellung aller Farben in jedem Ihrer Farbschemata. Sie können auch neue Schemata erstellen oder bereits existierende löschen. Jedes Schema hat Einstellungen für Farben sowie normale und hervorgehobene Textstile.

KatePart startet diese Seite mit dem aktuell aktiven Farbschema. Wenn Sie an einem anderen Farbschema Veränderungen vornehmen wollen, dann wählen Sie dieses mit im Auswahlfeld **Schema**. Mit den Knöpfen **Neu** und **Löschen** können Sie neue Schemata als Kopie eines vorhandenen Schemas erstellen oder vorhandene entfernen.

Unten auf dieser Seite wählen Sie das **Standardschema für `Anwendung`**.

Dies wird ausführlich in diesem Abschnitt Abschnitt 6.3.5 beschrieben.

7.1.3 Bearbeitungseinstellungen

7.1.3.1 Allgemein

Statischer Zeilenumbruch

Zeilenumbruch ist eine Funktion, die bewirkt, dass der Editor automatisch eine neue Textzeile beginnt und den Cursor an den Anfang dieser neuen Zeile verschiebt. Wenn diese Option aktiv ist, beginnt KatePart automatisch eine neue Zeile, sobald die aktuelle Zeile die Länge erreicht, die im Feld Zeilenumbruch bei: angegeben ist.

Statischen Zeilenumbruch aktivieren

Schaltet den statischen Zeilenumbruch ein und aus.

Markierung für statischen Zeilenumbruch anzeigen (falls zutreffend)

Wenn eingeschaltet, dann wird eine senkrechte Linie in der Spalte, an der der Zeilenumbruch erfolgt, angezeigt. Die Position wird in **Einstellungen** → **Editor einrichten ...** auf der Karte Bearbeitung festgelegt. Die Markierung wird nur dann angezeigt, wenn Sie eine Schrift mit fester Buchstabenbreite verwenden.

Zeilenumbruch bei:

Wenn die Option Statischen Zeilenumbruch aktivieren eingeschaltet ist, dann wird hier eingestellt, bei welcher Zeilenlänge in Zeichen der Editor automatisch eine neue Zeile beginnt.

Eingabemodus

Der hier ausgewählte Eingabemodus wird aktiviert, wenn eine neue Ansicht geöffnet wird. Sie können den VI-Eingabemodus weiterhin über das Menü **Bearbeiten** für jede Ansicht separat ein-/ausschalten.

Automatische Klammern

Wenn dies Option aktiv ist, setzt KatePart beim Eingeben einer linken Klammer ([, **(** oder {) automatisch eine rechte Klammer des gleichen Typs (}, **)**, or]) rechts vom Cursor. Diese braucht dann zum Schließen der Klammer nur noch übersprungen zu werden.

Ist Text ausgewählt, wird bei Eingabe dieser Zeichen der Text umgebrochen.

Kopieren und Einfügen

Die aktuelle Zeile kopieren/ausschneiden, wenn keine Markierung vorliegt

Ist diese Einstellung aktiv und kein Text ausgewählt, werden die Aktionen Kopieren und Ausschneiden für die ganze Textzeile an der aktuellen Cursorposition ausgeführt.

7.1.3.2 Textnavigation

Text-Cursor-Bewegung

Intelligente Tasten Pos 1 und Ende

Wenn dieses Feld angekreuzt ist, dann bewegt das Drücken der Taste Pos1 den Cursor an den Beginn des Textes in der aktuellen Zeile, Leerzeichen und Tabulatoren davor werden übersprungen.

Cursor folgt Bild auf/ab

Diese Option ändert das Verhalten des Cursors, wenn der Benutzer die Tasten **Bild auf** oder **Bild ab** drückt. Wenn diese Option ausgeschaltet ist, dann bleibt der Cursor an der gleichen Stelle innerhalb des sichtbaren Bildes, es wird also der Text unter dem Cursor verschoben. Bei Erreichen des Textendes oder Textanfangs kann dies aber nicht immer funktionieren. Bei eingeschalteter Option wird der Cursor beim ersten Drücken der Taste an den Bildanfang oder das Bildende bewegt. Erst beim nächsten Betätigen wird dann der Text bewegt.

Cursor-Bewegung mit Binnenmajuskeln (Camel Case) aktivieren

Diese Einstellung ändert das Verhalten des Cursors, wenn der Benutzer die Tastenkombination **Strg-Pfeil links** oder **Strg-Pfeil rechts** drückt. Wenn diese Einstellung nicht ausgewählt ist, springt der Textcursor über die ganzen Wörter. Wenn diese Einstellung ausgewählt ist, springt der Cursor zu den Binnenmajuskeln (Großbuchstaben innerhalb eines Wortes).

Automatische Cursor-Zentrierung:

Setzt die Anzahl der Zeilen, die der Cursor Abstand vom oberen oder unteren Bildrand hält, wenn möglich.

Textmarkierungsmodus

Normal

Die Auswahl wird durch Texteingaben überschrieben und geht beim Bewegen des Cursors verloren.

Beständig

Die Auswahl bleibt auch beim Bewegen des Cursors und bei Texteingaben bestehen.

Rollen über das Dokumentende hinaus zulassen

Mit dieser Einstellung ist es möglich, über das Dokumentende hinaus zu blättern. Damit kann das Ende des Dokuments im Fenster zentriert oder bis zum Anfang der Ansicht hochgeschoben werden.

Rücktaste löscht Basis- und zugehöriges diakritische Zeichen

Ist dies aktiviert, werden zusammengesetzte Zeichen mit den zugehörigen diakritischen Zeichen gelöscht, nicht nur die Basiszeichen allein. Dies ist nützlich für indische Schriften.

7.1.3.3 Einrückung

Standard-Einrückungsmodus:

Hier wählen Sie den Einrückungsmodus, den Sie als Standard benutzen wollen. Es wird empfohlen, dass Sie hier **`Kein`** oder **`Normal`** einstellen und die Einstellungen für Dateitypen benutzen, um andere Einrückungen, wie zum Beispiel C/C++-Quelltext oder XML zu wählen.

Einrücken mit

Tabulatoren

Wenn dieses Feld angekreuzt ist, setzt der Editor Tabulatorzeichen ein, wenn die Taste **Tab** gedrückt oder die Automatische Einrückung benutzt wird.

Leerzeichen

Wenn dieses Feld angekreuzt ist, setzt der Editor eine berechnete Anzahl von Leerzeichen ein, wenn die Taste **Tab** gedrückt oder Automatische Einrückung benutzt wird. Die Anzahl der Leerzeichen wird aus der Position im Text und der Einstellung für `Tabulatorweite` berechnet.

Tabulatoren und Leerzeichen

Ist diese Einstellung aktiv, werden Leerzeichen wie oben beschrieben eingefügt, wenn die **Tab**taste am Zeilenanfang gedrückt oder Einrückung benutzt wird. Wird die **Tab**taste mitten in der Zeile oder am Zeilenende gedrückt, werden Tabulatorzeichen eingefügt.

Tabulatorweite:

Hier wird die Anzahl der Leerzeichen angegeben, die für ein Tabulatorzeichen angezeigt werden.

Einrückungstiefe:

Die Einrückungstiefe ist die Anzahl Leerzeichen, die zum Einrücken einer Zeile verwendet wird. Ist das Einrücken mit Tabulator eingestellt , wird für die Einrückung ein Tabulator-Zeichen verwendet, sofern die Einrückungstiefe durch die Tabulatorweite teilbar ist.

Einrückungseigenschaften

Zusätzliche Leerzeichen beibehalten

Ist diese Einstellung nicht aktiv, richtet die Änderung der Einrückungsebene eine Zeile an einem Vielfachen der angegebenen **Einrückungstiefe** aus.

Einrückung von Text vornehmen, der aus der Zwischenablage eingefügt wird

Ist diese Einstellung ausgewählt, wird aus der Zwischenablage eingefügter Text eingerückt. Durch die Aktion **Rückgängig** kann die Einrückung rückgängig gemacht werden.

Einrückungs-Aktionen

Rücktaste verringert Einrückungsebene (im führenden Leerbereich einer eingerückten Zeile)

Ist diese Einstellung markiert, verringert die **Rücktaste** die Einrückungsebene, wenn der Cursor in den Leerzeichen am Anfang einer Zeile steht.

Aktion der Tabulator-Taste (wenn keine Markierung vorliegt)

Wenn Sie möchten, dass die **Tab**taste die aktuelle Zeile im aktuellen Quelltextblock wie in Emacs ausrichtet, weisen Sie der **Tab**taste den Kurzbefehl **Ausrichten** zu.

Immer zur nächsten Tabulatorposition vorrücken

Ist diese Einstellung aktiv, fügt die Tabulator-Taste immer Leerzeichen bis zum nächsten Tabulatorstop ein. Ist die Einstellung **Leerzeichen statt Tabulatoren für Einrückung verwenden** auf der Karteikarte **Allgemein** der Seite **Bearbeitung** aktiv, werden Leerzeichen eingefügt, anderenfalls ein einzelner Tabulator.

Einrückungsebene immer erhöhen

Ist diese Einstellung aktiv, fügt die **Tab**-Taste immer die unter **Einrückungstiefe** angegebene Anzahl Leerzeichen ein.

Einrückungsebene erhöhen, wenn im Leerzeichenbereich am Zeilenanfang

Ist die Einstellung markiert, rückt die Taste **Tab** entweder die aktuelle Zeile ein oder springt zur nächsten Tabulatorposition. Wird der Tabulator an oder vor der Position des ersten Zeichens eingefügt, dass kein Leerzeichen ist, oder liegt eine Markierung vor, wird die aktuelle Zeile um die Anzahl Zeichen eingerückt, die unter **Einrückungstiefe:** angegeben ist. Wird der Tabulator nach dem ersten Zeichen, dass kein Leerzeichen ist, eingefügt und es liegt keine Markierung vor, werden Leerräume bis zum Erreichen der nächsten Tabulatorposition eingefügt. Ist die Einstellung **Leerzeichen statt Tabulatoren für Einrückung verwenden** auf der Karteikarte **Allgemein** der Seite **Bearbeitung** aktiviert, werden Leerzeichen eingefügt, anderenfalls ein Tabulatorzeichen.

7.1.3.4 Autovervollständigung

Allgemein

Autovervollständigung aktivieren
Ist dies aktiviert, erscheint bei der Eingabe automatisch eine Liste mit Texteinträgen, mit denen der aktuelle Text unter dem Cursor vervollständigt werden kann.

Ersten Vervollständigungseintrag automatisch wählen
Wenn dies aktiviert ist, wird das erste Element der automatischen Vervollständigung immer vorausgewählt, so dass Sie es mit **Eingabe** einfügen können. Wenn Sie ein solches Verhalten nicht möchten, z. B. wenn Sie mit **Eingabe** nur einen Zeilenumbruch einfügen wollen, dann deaktivieren Sie diese Einstellung.

Minimale Wortlänge für Vervollständigung
Bei der Texteingabe sucht die Wortvervollständigung im Dokument nach Wörtern, die mit dem bereits eingegebenen Text beginnen. Diese Einstellung legt die minimale Anzahl der einzugebenden Zeichen fest, ab der die Wortvervollständigung aktiviert und das Feld mit passenden Vorschlägen angezeigt wird.

Bei Vervollständigung Wortende entfernen
Entfernt ein bestehendes Wortende, wenn eine Vervollständigung aus der Liste gewählt wird

Schlüsselwortvervollständigung
Die eingebaute automatische Vervollständigung verwendet die Schlüsselwörter, die in der Syntaxhervorhebung definiert sind.

7.1.3.5 Rechtschreibprüfung

Die Einstellungen für die Rechtschreibprüfung werden Sie im Systemeinstellungen-Module Rechtschreibprüfung erläutert.

7.1.3.6 VI-Eingabemodus

Allgemein

VI-Befehle überschreiben Kate-Kurzbefehle
Wenn diese Einstellung aktiviert ist, werden VI-Befehle KatePart's eingebaute Befehle überschreiben. Beispielsweise wird **Strg**+**R** eine Aktion wiederherstellen anstatt die Standard-Aktion auszuführen (den Dialog „Suchen und Ersetzen“ anzeigen).

Relative Zeilennummern anzeigen
Ist dies aktiviert, wird immer die aktuelle Zeile als Zeile „0“ gezählt und Zeilen über und unter der aktuellen Zeile relativ zur aktuellen Zeile nummeriert.

Tastenzuordnung
Mit der Tastenzuordnung können Sie die Bedeutung von gedrückten Tasten auf der Tastatur anpassen. Sie können Befehle auf andere Tasten umlegen oder besondere Tastenkombinationen definieren, um eine Serie von Befehlen auszuführen.

Beispiel:

F2 -> **`I--` Esc**
Dadurch wird einer Zeile beim Drücken von **F2** die Zeichenfolge **`I--`** vorangestellt.

7.1.4 Öffnen/Speichern

7.1.4.1 Allgemein

Dateiformat

Kodierung

Hier wird die Standardkodierung zum Öffnen/Speichern von Dateien festgelegt, falls diese nicht im Öffnen-/Speichern-Dialog oder über die Befehlszeile bereits festgelegt ist.

Erkennung der Kodierung

Wählen Sie einen Eintrag aus der Liste im Auswahlfeld, um die automatische Erkennung abzuschalten oder mit **Allgemein** für alle Kodierungen zu aktivieren. Da diese Einstellung oft nur die Kodierung utf-8 oder utf-16 erkennt, wird bei der Auswahl einer Region mit dafür angepassten Verfahren die richtige Kodierung eher erkannt. Falls weder die oben angegebene Kodierung, noch die im Öffnen-/Speichern-Dialog oder die über die Befehlszeile angegebene Kodierung für die Datei passend sind, wird die automatische Erkennung gestartet.

Ausweich-Kodierung:

Hier wird die Ausweich-Kodierung festgelegt, mit der Dateien geöffnet werden, falls keine der sonstigen angegebenen Kodierungen passend ist. Bevor die Ausweich-Kodierung eingesetzt wird, wird zunächst versucht, die korrekte Kodierung anhand einer Byte-Reihenfolge-Markierung am Anfang der Datei automatisch festzustellen: Wenn eine gefunden wird, wird die korrekte Unicode-Kodierung verwendet; ansonsten wird die Kodierungserkennung gestartet. Erst wenn beides fehlschlägt, wird die Ausweich-Kodierung verwendet.

Zeilenende

Wählen Sie den Zeilenendemodus für das aktuelle Dokument. Sie haben die Auswahl zwischen UNIX®, DOS/Windows® oder Macintosh.

Automatische Zeilenendeerkennung

Wenn dieses Feld angekreuzt ist, dann stellt der Editor den Zeilenendetyp automatisch fest. Dazu wird das erste gefundene Zeilenende benutzt.

Byte-Reihenfolge-Markierung aktivieren (BOM)

Die Byte-Reihenfolge-Markierung ist eine spezielle Abfolge am Anfang von Unicode-kodierten Dokumenten. Sie unterstützt Editoren beim Öffnen von Textdokumenten mit der richtigen Unicode-Kodierung. Die Byte-Reihenfolge-Markierung ist im angezeigten Dokument nicht sichtbar. Weitere Informationen finden Sie im Artikel Byte-Reihenfolge-Markierung.

Begrenzung der Zeilenlänge

Wegen Mängeln in Qt™ verarbeitet KatePart sehr lange Zeilen nur mit eingeschränkter Leistungsfähigkeit. Daher werden Zeilen mit einer größeren Anzahl von Zeichen als hier angegeben automatisch umgebrochen. Um den automatischen Umbruch abzuschalten, setzen Sie diesen Wert auf `0`.

Automatische Bereinigung beim Speichern

Leerzeichen am Zeilenende entfernen

Der Editor entfernt überflüssige Leerzeichen an den Zeilenenden beim Speichern.Sie können **Nie** zum Abschalten dieser Funktion, **Nur geänderte Zeilen** oder **Im gesamten Dokument** einstellen und so die Anwendung dieser Funktion steuern.

Beim Speichern Zeilenumbruch am Ende der Datei einfügen

Der Editor fügt beim Speichern automatisch ein Zeilenvorschubzeichen am Ende der Datei an, wenn noch keins vorhanden ist.

7.1.4.2 Erweitert

Sicherungskopie beim Speichern

Sicherungskopie beim Speichern weist KatePart an, vor dem Speichern von Dateien eine Sicherungskopie unter: <Präfix><Dateiname><Erweiterung>' zu erstellen. Die Erweiterung ist standardmäßig ~ und der Präfix ist standardmäßig leer.

Lokale Dateien
Wenn dieses Feld angekreuzt ist, werden von lokalen Dateien Sicherungskopien erstellt.

Dateien auf Fremdrechnern
Wenn dieses Feld angekreuzt ist, werden von auf Fremdrechnern bearbeiteten Dateien Sicherungskopien erstellt.

Präfix
Geben Sie hier den Präfix ein, der dem Dateinamen der Sicherungskopie vorangestellt wird.

Erweiterung
Geben Sie hier die Erweiterung ein, die an den Dateinamen der Sicherungskopie angehängt wird.

Swap-Dateieinstellungen
KatePart ist in der Lage, große Teile dessen, was seit der letzten Sicherung geschrieben wurde, bei einem Absturz oder einem Stromausfall wiederherzustellen. Nach der ersten Veränderung des aktuellen Dokumentes wird eine Swap-Datei (.swp.<filename>) erzeugt. Wenn der Nutzer die Änderungen nicht speichert und KatePart abstürzt, bleibt die Swap-Datei auf der Festplatte. Beim Öffnen eines Dokumentes prüft KatePart, ob eine Swap-Datei zu diesem Dokument existiert und wenn das der Fall ist, dann fragt KatePart, ob die verlorenen Änderungen wiederhergestellt werden sollen. Dabei kann der Nutzer diese Änderungen ansehen. Die Swap-Datei wird bei jedem Sichern und beim normalen Beenden von KatePart gelöscht.
KatePart gleicht die offenen Dateien mit den Swap-Dateien auf der Festplatte alle 15 Sekunden ab, aber nur wenn diese seit dem letzten Abgleich geändert wurden. Der Nutzer kann diesen Abgleich durch Ankreuzen von **Deaktivieren**abschalten, das kann aber zu Datenverlust führen.
Ist dies aktiviert, dann werden die Swap-Dateien im Ordner der Datei gespeichert. Mit **Alternativer Ordner** können Sie einen bestimmten Ordner für die Swap-Dateien angeben. Dies sollte bei Netzwerkwerk-Dateisystemen benutzt werden, um unnötige Netzwerkbelastungen zu vermeiden.

7.1.4.3 Modi & Dateitypen

Diese Seite dient zur Einstellung von abweichenden Einstellungen für Dokumente bestimmter MIME-Typen. Wenn ein Dokument in den Editor geladen wird, dann versucht dieser einen schon festgelegten Datentyp zu finden, auf den die Merkmale eines MIME-Typs passen und verwendet dann die Variablen, die für diesen Datentyp festgelegt wurden. Wenn mehrere Datentypen passend sind, dann wird der Typ verwendet, der die höchste Priorität besitzt.

Dateityp:

Der Dateityp mit der höchsten Priorität wird im ersten Auswahlfeld angezeigt. Wenn mehrere Dateitypen gefunden wurden, werden diese ebenfalls aufgelistet.

Neu
Dieser Knopf wird zum Erstellen eines neuen Dateityps benutzt. Wenn Sie diesen Knopf drücken, werden die Inhalte aller Felder hierunter gelöscht und Sie können die gewünschten Eigenschaften für den neuen Dateityp dort eintragen.

Löschen
Um einen existierenden Dateityp zu entfernen, klicken Sie auf den Knopf **Löschen**.

Eigenschaften des *aktuellen Dateityps*

Der Dateityp mit der höchsten Priorität wird im ersten Auswahlfeld angezeigt. Wenn mehrere Dateitypen gefunden wurden, werden diese ebenfalls aufgelistet.

Name:
Geben Sie hier einen aussagekräftigen Namen an, der dann im Menü **Extras** → **Dateityp** erscheint.

Abschnitt:
Der Abschnittsname wird zum Organisieren der vielen Dateitypen in Menüs benutzt. Geben Sie hier einen aussagekräftigen Namen an, der dann im Menü **Extras** → **Dateityp** als Untermenü erscheint.

Variablen:
Dieser Eintrag erlaubt das Einstellen von KateParts Optionen für die Dateien dieses MIME-Typs unter Benutzung der Variablen von KatePart. Sie können so fast alle Einstellungen wie zum Beispiel Hervorhebungen, Einrückung usw. einstellen.

Drücken Sie auf das Symbol rechts neben dem Eingabefeld. dann wird eine Liste aller vorhandenen Variablen und deren Beschreibung angezeigt. Klicken Sie auf das Ankreuzfeld links, um eine bestimmte Variable zu aktivieren und stellen Sie dann rechts den Wert der Variablen ein. Für einige Variablen gibt es Auswahlfelder mit zulässigen Werten, für andere Variablen müssen Sie die Werte direkt eingeben.

Weitere Informationen zu diesen Variablen finden Sie unter Einstellungen mit Dokumentvariablen.

Hervorhebung:
Wenn Sie einen neuen Dateityp erstellen, können Sie in diesem Auswahlfeld einen Dateityp für die Hervorhebung auswählen.

Einrückungsmodus:
In diesem Auswahlfeld kann der Einrückungsmodus für neue Dokumente eingestellt werden.

Dateierweiterungen:
Das Feld Dateierweiterungen erlaubt das Auswählen von Dateien nach dem Dateinamen. Ein typischer Eintrag hier besteht aus einem Stern und der Dateinamenserweiterung, zum Beispiel `*.txt; *.text`. Tragen Sie hier mehrere Typen ein, werden diese Einträge durch Semikolons getrennt.

MIME-Typen:
Zeigt ein Dialogfeld an, in dem Sie einfach und schnell MIME-Typen auswählen können.

Priorität:
Stellen Sie hier die Priorität für den Dateityp ein. Wenn auf ein Dokument mehrere Dateitypen zutreffen, wird der Typ mit der höchsten Priorität benutzt.

7.2 Einstellungen mit Dokumentvariablen

KatePart Variablen sind KatePart Dokumentvariablen, ähnlich der Modelines in Emacs und Vi. In Katepart haben die Dokumentvariablen das folgende Format: **`kate: VARIABLENAME VALUE ; [ VARIABLENAME VALUE; ... ]`**. Die Zeilen können natürlich auch in einem Kommentar stehen, wenn das Format des Dokuments Kommentare beinhaltet. Variablennamen sind einzelne Wörter ohne Zwischenräume und alles bis zum nächsten Semikolon sind Werte. Das Semikolon ist vorgeschrieben.

Hier ein Beispiel für eine Variablenzeile, die die Einrückung für Quelltext in C++, Java™ oder JavaScript einschaltet:

```
// kate: replace-tabs on; indent-width 4; indent-mode cstyle;
```

ANMERKUNG
Nur die ersten und letzten 10 Zeilen eines Dokuments werden nach Dokumentvariablen durchsucht.

Zusätzlich können Dokumentvariablen in eine Datei mit dem Namen `.kateconfig` in jedem beliebigen Ordner geschrieben werden. Die Einstellungen dieser Dokumentvariablen werden so verwendet, als wenn sie als Modelines in jeder Datei im Ordner und allen Unterordnern eingefügt wären. Dokumentvariablen in `.kateconfig` verwenden die gleiche Syntax wie Modelines, aber mit zusätzlichen Optionen.

Es gibt Variablen für fast alle Einstellungen in KatePart. Außerdem können Module Variablen benutzen. In diesem Fall sind sie in der Dokumentation der Module dokumentiert.

KatePart kann Einstellungen aus `.editorconfig`-Dateien einlesen, wenn die Bibliothek editorconfig installiert ist. KatePart sucht immer automatisch nach einer Datei mit dem Namen `.editorconfig`, wenn Sie ein Dokument öffnen. Allerdings werden die Einstellungen aus `.kateconfig`-Dateien zuerst benutzt.

7.2.1 Wie KatePart Variablen benutzt

Beim Einlesen der Einstellungen werden von katepart

- die globalen Einstellungen,
- optionale Daten zur aktuellen Sitzung,
- die Einstellungen zum „Dateityp”,
- Dokumentvariablen in `.kateconfig`,
- Variablen im Dokument selbst,
- Einstellungen während der aktuellen Sitzung über das Menü oder die Befehlszeile

in der angegebenen Reihenfolge gelesen und angewendet. Wie Sie sehen, werden Dokumentvariablen nur durch Änderungen zur Laufzeit überschrieben. Immer wenn ein Dokument gespeichert wird, werden die Dokumentvariablen neu eingelesen und überschreiben dann von der Befehlszeile oder über das Menü vorgenommene Einstellungsänderungen.

Jede hier nicht beschriebene Variable ist im Dokument gespeichert und kann durch andere Objekte wie Erweiterungen abgefragt werden, die diese Variablen für ihre eigenen Zwecke setzen können. Zum Beispiel nutzt der Modus für die Variablenbasierte-Einrückung Dokumentvariablen zum Speichern der Einstellungen.

Die hier beschriebenen Variablen sind in KatePart Version 5.38 enthalten. Es werden in der Zukunft sicher weitere Variablen hinzugefügt werden. Es gibt drei Typen von Variablen mit den folgenden gültigen Werten:

- BOOL - on | off | true | false | 1 | 0
- INTEGER - eine ganze Zahl
- STRING - alles andere

7.2.2 Verfügbare Variablen

`auto-brackets` [BOOL]
Automatischen Einfügen von Klammern aktivieren.

`auto-center-lines` [INT]
Setzt die Anzahl der automatisch zentrierten Zeilen.

`background-color` [STRING]
Setzt die Hintergrundfarbe des Dokuments. Der Wert muss als gültige Farbe ausgewertet werden können, also z. B. **`#ff0000`**.

`backspace-indents` [BOOL]
Schaltet die Verringerung des Einrückens beim Drücken der Taste **Rücktaste** ein oder aus.

`block-selection` [BOOL]
Schaltet die Blockauswahl ein und aus.

`bom | byte-order-mark | byte-order-marker` [BOOL]
Schaltet die Markierung für die Bytereihenfolge (BOM) ein und aus, wenn Dokumente in einem Unicodeformat (utf8, utf16, utf32) gespeichert werden.

Ab Version: Kate 3.9 (KDE 4.9)

`bracket-highlight-color` [STRING]
Setzt die Hintergrundfarbe für die Hervorhebung von Klammern. Der Wert muss als gültige Farbe ausgewertet werden können, also z. B. **`#ff0000`**.

`current-line-color` [STRING]
Setzt die Farbe für die aktuelle Zeile. Der Wert muss als gültige Farbe ausgewertet werden können, also z. B. **`#ff0000`**.

`default-dictionary` [STRING]
Legt das Standardwörterbuch für die Rechtschreibprüfung fest.

Ab Version: Kate 3.9 (KDE 4.9)

`dynamic-word-wrap` [BOOL]
Schaltet den dynamischen Zeilenumbruch ein und aus.

`eol | end-of-line` [STRING]
Setzt das Format für das Zeilenende. Gültige Werte hierfür sind: „unix", „mac" und „dos".

`folding-markers` [BOOL]
Schaltet die Anzeige von Quelltextausblendungen ein und aus.

`folding-preview` [BOOL]
Vorschau der Text-Ausblendung am Editor-Rand anzeigen

`font-size` [INT]
Setzt die Schriftgröße des Dokuments.

`font` [STRING]
Setzt die Schriftart des Dokuments . Der Wert muss eine gültige Schriftart bezeichnen, also z. B. **`courier`**.

`hl | syntax` [STRING]
Setzt den Hervorhebungsmodus. Es können alle Namen, die auch in den Menüs vorhanden sind, verwendet werden. z.B für C++ benutzen Sie einfach **`C++`**.

icon-bar-color [STRING]

Setzt die Farbe des Symbolrandes. Der Wert muss als eine gültige Farbe übersetzt werden können, also z. B. **#ff0000**.

icon-border [BOOL]

Schaltet die Anzeige des Symbolrandes ein und aus.

indent-mode [STRING]

Setzt den Modus für das automatische Einrücken. Die Einstellungen **normal**, **cstyle**, **haskell**, **lilypond**, **lisp**, **python**, **ruby** und **xml** sind möglich. Sehen Sie unter Automatisches Einrücken benutzen für Einzelheiten nach.

indent-pasted-text [BOOL]

Aktiviert/deaktiviert die Anpassung der Einrückung von Text, der aus der Zwischenablage eingefügt wird

Ab Version: Kate 3.11 (KDE 4.11)

indent-width [INT]

Setzt die Breite der Einrückung.

keep-extra-spaces [BOOL]

Legt fest, ob zusätzliche Leerzeichen bei der Berechnung der Einrückungweite beibehalten werden.

line-numbers [BOOL]

Schaltet die Anzeige der Zeilennummern ein und aus.

newline-at-eof [BOOL]

Fügt beim Speichern des Dokuments eine leere Zeile am Ende der Datei (EOF) an.

Ab Version: Kate 3.9 (KDE 4.9)

overwrite-mode [BOOL]

Schaltet den Überschreibmodus ein und aus.

persistent-selection [BOOL]

Schaltet die durchgehende Auswahl ein und aus.

replace-tabs-save [BOOL]

Schaltet das Ersetzen von Tabulatoren durch Leerzeichen beim Speichern des Dokuments ein und aus.

replace-tabs [BOOL]

Schaltet das sofortige Ersetzen von Tabulatoren durch Leerzeichen ein und aus.

remove-trailing-spaces [STRING]

Entfernt Leerzeichen am Zeilenende beim Speichern des Dokuments. Gültige Optionen sind:

- **none**, **–** or **0**: Leerzeichen am Zeilenende nie entfernen.
- **modified**, **mod**, **+** or **1**: Leerzeichen am Zeilenende nur in geänderten Zeilen entfernen. Diese geänderten Zeilen werden durch das Zeilenänderungssystem gekennzeichnet.
- **all**, ***** or **2**: Leerzeichen am Zeilenende im gesamten Dokument entfernen.

scrollbar-minimap [BOOL]

Textgrafik auf Bildlaufleiste anzeigen

scrollbar-preview [BOOL]

Vorschau an Bildlaufleiste anzeigen.

scheme [ZEICHENFOLGE]

Setzt das Farbschema von Kate. Die Zeichenfolge muss ein gültiger Name für ein Farbschema sein, sonst wird diese Einstellung ignoriert.

selection-color [STRING]

Setzt die Farbe für ausgewählten Text. Der Wert muss als gültige Farbe ausgewertet werden können, also z. B. **#ff0000**.

show-tabs [BOOL]

Schaltet die Anzeige von Tabulatorzeichen ein und aus.

smart-home [BOOL]

Schaltet die intelligente Funktion der Tasten Pos1 und Ende ein oder aus.

tab-indents [BOOL]

Schaltet das Einrücken mit der **Tab**taste ein und aus.

tab-width [INT]

Setzt die angezeigte Weite für ein Tabulatorzeichen.

undo-steps [INT]

Setzt die Anzahl der gespeicherten Schritte für die Funktion Rückgängig.

Anmerkung: Ab Version Kate 3 in KDE4 wird diese Variable ignoriert. Die maximale Anzahl von Schritten für Rückgängig ist unbegrenzt.

word-wrap-column [INT]

Setzt die Zeilenlänge für den Statischen Zeilenumbruch.

word-wrap-marker-color [STRING]

Setzt die Farbe für Zeilenumbruchmarkierungen. Der Wert muss als gültige Farbe ausgewertet werden können, also z. B. **#ff0000**.

word-wrap [BOOL]

Schaltet den statischen Zeilenumbruch ein und aus.

7.2.3 Zusätzliche Optionen in `.kateconfig`-Dateien

KatePart sucht nach einer `.kateconfig`-Datei nur in lokalen Dateien, nicht in Dateien auf anderen Rechnern. Außerdem können Optionen für Platzhalter (Dateierweiterungen) wie folgt eingestellt werden:

```
kate: tab-width 4; indent-width 4; replace-tabs on;
kate-wildcard(*.xml): indent-width 2;
kate-wildcard(Makefile): replace-tabs off;
```

In diesem Beispiel wird für alle Dateien eine Tabulatorweite von vier Leerzeichen, eine Einrückungstiefe von vier Leerzeichen verwendet und Tabulatoren werden durch Leerzeichen ersetzt. Bei allen `*.xml`-Dateien wird jedoch eine Einrückungstiefe von zwei Leerzeichen benutzt., außerdem in Make-Dateien nur Tabulatoren, d. h. sie werden nicht durch Leerzeichen ersetzt.

Platzhalter werden durch Semikolon getrennt, d. h. Sie können auch mehrere Erweiterungen wie im nächsten Beispiel angeben:

```
kate-wildcard(*.json;*.xml): indent-width 2;
```

Weiterhin können Sie MIME-Typen auch zur Erkennung bestimmter Dateien benutzen. Um z. B. alle Dateien mit C++-Quelltexten mit vier Leerzeichen ein zu rücken, verwenden Sie:

```
kate-mimetype(text/x-c++src): indent-width 4;
```

ANMERKUNG

Außer in `.kateconfig`-Dateien können Dokumentvariablen mit Platzhaltern und MIME-Typen auch in Dateien direkt als Kommentare benutzt werden.

Kapitel 8

Danksagungen und Lizenz

Mitarbeit:

- Christoph Cullmann cullmann@kde.org
- Michael Bartl michael.bartl1@chello.at
- Phlip phlip_cpp@my-deja.com
- Anders Lund anders@alweb.dk
- Matt Newell newellm@proaxis.com
- Joseph Wenninger kde@jowenn.at
- Jochen Wilhelmy digisnap@cs.tu-berlin.de
- Michael Koch koch@kde.org
- Christian Gebauer gebauer@kde.org
- Simon Hausmann hausmann@kde.org
- Glen Parker glenebob@nwlink.com
- Scott Manson sdmanson@altel.net
- John Firebaugh jfirebaugh@kde.org
- Nibaldo González nibgonz@gmail.com

Die Dokumentation zu KatePart basiert auf der Originaldokumentation zu KWrite angepasst für alle Nutzer von KatePart.

Die Originaldokumentation für KWrite wurde von Thad McGinnis ctmcginnis@compuserve.com verfasst, viele Änderungen stammen von Christian Tibirna tibirna@kde.org. Kontrollgelesen und nach Docbook konvertiert wurde diese von Lauri Watts lauri@kde.org. Aktualisierungen kamen von Anne-Marie Mahfouf annma@kde.org und Anders Lund anders@alweb.dk.

Die aktuelle Dokumentation für KatePart wird von T.C. Hollingsworth tchollingsworth@gmail.com gepflegt. Bitte schicken Sie Kommentare oder Verbesserungsvorschläge unter der Adresse kwrite-devel@kde.org an die Entwickler-Mailingliste von KatePart oder eröffnen Sie ein Ticket im KDE Bugtracking System.

Übersetzungen von:

- Thomas Diehl thd@kde.org, GUI-Übersetzung
- Matthias Schulzmatthias.schulz@kdemail.net, Übersetzung der Dokumentation

Kapitel 9

Der VI-Eingabemodus

Erlend Hamberg
Übersetzung: Frederik Schwarzer

9.1 VI-Eingabemodus

Ziel des VI-Modus ist nicht, Vim zu ersetzen indem alle Vim-Funktionen unterstützt werden. Das Ziel ist es, die „Vim-Art" der Textbearbeitung und somit die angelernten Gewohnheiten in Programmen zur Verfügung zu stellen, die den KatePart-Texteditor als ihren internen Editor verwenden.
Der VI-Modus hat zum Ziel, sich in die Programme zu integrieren und, wo sinnvoll, das Verhalten von Vim nachzubilden. Zum Beispiel öffnet `:w` in KateParts VI-Modus einen Dialog zum Speichern.

Um den VI-Modus für alle neuen Ansichten zu aktivieren, gehen Sie auf **Einstellungen** → **KatePart einrichten ...**+**Bearbeitung** → **VI-Eingabemodus**. Auf dieser Karteikarte können Sie den VI-Modus einrichten und die Tastenzuordnungen für diesen Modus anlegen und ändern. Der VI-Modus kann auch mit dem Menüpunkt **VI-Eingabemodus** im Menü **Bearbeiten** ein- bzw. ausgeschaltet werden. (Der Standardkurzbefehl ist **Meta**+**Strg**+**V**, wobei **Meta** normalerweise die **Windows**-Taste ist.)

> ANMERKUNG
> Viele Kurzbefehle im VI-Modus beachten die Groß- und Kleinschreibung, im Gegensatz zu den meisten KDE-Kurzbefehlen. Das heißt, dass `y` und `Y` verschiedene Kurzbefehle sind. Um den Befehl `y` (kopieren) einzugeben, überprüfen Sie dass die **Feststelltaste** nicht aktiviert ist und drücken **Y** auf der Tastatur. Um den Befehl `Y` (kopieren bis zum Zeilenende) einzugeben, drücken Sie die Tastenkombination **Umschalt**+**Y**.
> Die betrifft nicht die Kurzbefehle mit der **Strg**-Taste, die unabhängig vom Status der **Feststelltaste** und ohne Drücken der **Umschalt**-Taste eingegeben werden können. Bei einige Befehle jedoch muss bei einer Tasteneingabe nach der Tastenkombination mit der **Strg**-Taste die Groß-/Kleinschreibung berücksichtigt werden. Um zum Beispiel den Befehl „**Strg**+**W**, `h`" (Wechsel zum rechten Fenster der geteilten Ansicht) einzugeben, überprüfen Sie dass die **Feststelltaste** nicht aktiviert ist, drücken die Tastenkombination **Strg**+**W** und dann **H**.

9.1.1 Inkompatibilitäten mit Vim

Es gibt ein paar Funktionen in KateParts VI-Modus, die mit Vim nicht kompatibel sind (abgesehen von den fehlenden Funktionen). Diese sind hier aufgelistet, einschließlich. der entsprechen-

den Begründungen.

- KatePart: **U** und **Strg**+**R** ist Wiederherstellen.

  Vim: **Strg**+**R** ist normales Wiederherstellen; **U** macht alle Änderungen in einer Zeile rückgängig.

  Der Grund dafür, in KatePartsVI-Modus **U** für die Wiederherstellen-Aktion zu verwenden, ist, dass der Kurzbefehl **Strg**+**R** voreingestellt von KatePartsErsetzen-Funktion belegt ist (Suchen und ersetzen). Der VI-Modus überschreibt keine KatePart-Kurzbefehle (dies kann in **Einstellungen** → **KatePart einrichten ...**+**Bearbeitung** → **Vi-Eingabemodus** eingestellt werden), weshalb eine Wiederherstellen-Aktion auch über einen „normalen" Tastendruck verfügbar sein muss. Davon abgesehen lässt sich die Funktionsweise des **U**-Befehls aus Vim nicht gut auf das interne System zum Rückgängigmachen in KatePart abbilden, weshalb es nicht einfach wäre, dies zu unterstützen.

- KatePart: der Befehl **print** öffnet den Dialog **Drucken**.

  Vim: der Befehl **print** gibt die Zeilen des angegebenen Bereichs wie sein Vorläufer ed aus.

  Befehle wie **:print** sind nicht nur im VI-Modus, sondern für alle KatePart-Benutzer verfügbar. Daher öffnet der **:print**-Befehl den bekannten Dialog zum Drucken, anstatt das Verhalten von Vim nachzubilden.

- KatePart: **Y** kopiert bis zum Ende der Zeile.

  Vim: **Y** kopiert gesamte Zeile, genau wie **yy**.

  Das Verhalten des **Y**-Befehls von VI kann als Fehlerhaft angesehen werden. Beim Ändern und Löschen wirken sich **cc**/ **dd** auf die gesamte Zeile aus und **C**/**D** arbeiten von der aktuellen Cursor-Position bis zum Zeilenende. Beide, **yy** und **Y** hingegen kopieren die gesamte Zeile. In KatePartsVI-Modus kopiert **Y** bis zum Zeilenende kopieren. Dieses Verhalten wird in der Vim-Dokumentation als „logischer" beschrieben.

- KatePart: **O** und **o** öffnen [*eine Anzahl*] neuer Zeilen und wechseln in den Eingabemodus.

  Vim: **O** und **o** öffnen eine neue Zeile und fügen den eingegebenen Text [*so oft*] ein, sobald der Eingabemodus verlassen wird.

  Diese unterscheiden sich hauptsächlich, weil sich viele Benutzer im Vim-IRC-Kanal (#vim auf Libera Chat) verwirrt über die Funktionsweise geäußert haben.

9.1.2 Wechseln der Modi

- Im *Normalen Modus* können Sie Befehle zum Navigieren und Ändern eines Dokuments eingeben. Dieser Modus ist der Standardmodus. Sie können aus allen anderen Modi mit der **Esc**-Taste in diesen Modus zurückkehren.
- Im *Visuellen Modus* können Sie Text in einem Dokument markieren. Die meisten Befehle aus dem Normalen Modus sind auch in diesem Modus gültig. Zum zeichenweise auszuwählen, wechseln in diesen Modus, indem Sie die Taste **v** drücken; für zeilenweise Auswahl, drücken Sie **V**.
- Im *Eingabemodus* können Sie das Dokument direkt bearbeiten. Sie wechseln in diesen Modus, indem Sie die Taste **i** oder einen der anderen, oben genannten Befehle drücken.
- Der *Befehlsmodus* ruft KatePartsBefehlszeile auf. Hier können Sie viele in Vi implementierte Befehle wie auch spezielle Befehle für KatePart aufrufen. Weitere Informationen über diese Befehle finden Sie unter Abschnitt 5.2. Um diesen Modus zu benutzen, drücken Sie die Taste **:**, geben den Befehl ein und drücken dann die **Eingabe**taste.

9.1.3 Einbindung in Kate's Funktionen

- Es wird automatisch in den Visuellen Modus gewechselt, wenn Text mit der Maus ausgewählt ist. Dieser Wechsel findet auch dann statt, wenn Funktion von Kate benutzt werden, die Text auswählen, wie zum Beispiel **Alles auswählen** aus dem Menü oder mit dem Kurzbefehl **Strg**+**A**.
- Marker in Vi und Lesezeichen in Kate sind integriert. Erstellen Sie einen Marker im Vi-Modus, dann wird auch das zugehörige Lesezeichen für Kate erstellt und im Menü **Lesezeichen** angezeigt. Umgekehrt wird mit einem Lesezeichen in Kate auch der zugehörige Marker in Vi an Spalte Null der Zeile erzeugt.

9.1.4 Unterstützte Befehle im normalen/visuellen Modus

a	Wechselt zum Eingabemodus und fügt hinter dem Cursor ein
A	Wechselt zum Eingabemodus und fügt hinter der Zeile ein
i	Wechselt zum Eingabemodus und fügt vor dem Cursor ein
Einfg	Wechselt zum Eingabemodus und fügt vor dem Cursor ein
I	Wechselt zum Einfügemodus und fügt vor dem ersten nicht leeren Zeichen auf der Zeile ein
gi	Wechselt zum Einfügemodus und fügt vor der Stelle ein, an der der letzte Einfügemodus verlassen wurde
v	Wechsel in den visuellen Modus; Auswahl von Zeichen
V	Wechsel in den visuellen Modus; Auswahl von Zeilen
Strg+**v**	Wechselt in den visuellen Modus; Auswahl von Blöcken
gb	Wechselt in den visuellen Modus und aktiviert die letzte Auswahl erneut
o	Fügt unter der aktuellen Zeile eine neue Zeile ein
O	Fügt über der aktuellen Zeile eine neue Zeile ein
J	Zeilen zusammenführen
c	Ändern: gefolgt von einer Richtungstaste; löscht ein Zeichen und wechselt in den Eingabemodus
C	Bis zum Zeilenende ändern: löscht bis zum Zeilenende und wechselt in den Eingabemodus
cc	Zeile ändern: Zeile löschen und in den Eingabemodus wechseln
s	Zeichen ersetzen
S	Zeilen ersetzen
dd	Zeile löschen

d	Gefolgt von einer Richtungstaste, um eine Zeile zu löschen
D	Löschen bis Zeilenende
x	Zeichen rechts vom Cursor löschen
Entf	Zeichen rechts vom Cursor löschen
X	Zeichen links vom Cursor löschen
gu	Gefolgt von einer Richtungstaste, um ein Zeichen in Kleinbuchstaben zu ändern
guu	Ändert die aktuelle Zeile in Kleinschreibung
gU	Gefolgt von einer Richtungstaste, um ein Zeichen in Großbuchstaben zu ändern
gUU	Ändert die aktuelle Zeile in Großschreibung
y	Gefolgt von einer Richtungstaste, um eine Zeile zu kopieren
yy	Zeile kopieren
Y	Zeile kopieren
p	Hinter dem Cursor einfügen
P	Vor dem Cursor einfügen
]p	Hinter dem Cursor mit Einrückung einfügen
[p	Vor dem Cursor mit Einrückung einfügen
r	Gefolgt von einem Zeichen, um das Zeichen hinter dem Cursor zu ersetzen
R	Zu Ersetzungsmodus wechseln
:	Zu Befehlsmodus wechseln
/	Suchen
u	Rückgängig
Strg+**R**	Wiederherstellen
U	Wiederherstellen
m	Marker setzen (kann später zum Navigieren verwendet werden)
n	Weitersuchen
N	Frühere suchen
>>	Zeile einrücken
<<	Zeileneinrückung rückgängig
>	Zeilen einrücken
<	Einrückung mehrere Zeilen rückgängig
Strg+**F**	Seite nach unten
Strg+**B**	Seite nach oben
ga	ASCII-Wert des Zeichens ausgeben
.	Letzte Änderung wiederholen
==	commandAlignLine
=	commandAlignLines
~	Groß-/Kleinschreibung des aktuellen Zeichens ändern
Strg+**S**	Ansicht waagerecht teilen
Strg+**V**	Ansicht senkrecht teilen
Strg+**W**, **w**	Wechselt zum nächsten Fenster der geteilten Ansicht
Strg+**W**, **h** **StrgW Pfeil links**	Wechselt zum linken Fenster der geteilten Ansicht

Strg+**W**, **l** **StrgW Pfeil rechts**	Wechselt zum rechten Fenster der geteilten Ansicht
Strg+**W**, **k** **StrgW Pfeil hoch**	Wechselt zum oberen Fenster der geteilten Ansicht
Strg+**W**, **j** **StrgW Pfeil runter**	Wechselt zum unteren Fenster der geteilten Ansicht

9.1.5 Unterstützte Richtungstasten

Diese können zum Bewegen in einem Dokument im Normalen und im Visuellen Modus oder im Zusammenspiel mit einem der oben genannten Befehle verwendet werden. Es können Nummern angehängt werden, um anzugeben, wie oft die entsprechende Bewegung ausgeführt werden soll.

h	Links
Pfeil links	Links
Rücktaste	Links
j	Nach unten
Pfeil runter	Nach unten
k	Nach oben
Pfeil hoch	Nach oben
l	Rechts
Pfeil rechts	Rechts
Leertaste	Rechts
$	Zum Zeilenende
Ende	Zum Zeilenende
0	Zum ersten Zeichen der Zeile (Spalte 0)
Pos 1	Zum ersten Zeichen der Zeile
^	Erstes Zeichen, das kein Leerzeichen ist, in dieser Zeile
f	Gefolgt von einem Zeichen rechts vom Cursor, zu dem der Cursor bewegt werden soll
F	Gefolgt von einem Zeichen links vom Cursor, zu dem der Cursor bewegt werden soll
t	Gefolgt von einem Zeichen rechts vom Cursor, vor das der Cursor bewegt werden soll
T	Gefolgt von einem Zeichen links vom Cursor, vor das der Cursor bewegt werden soll
gg	Zur ersten Zeile
G	Zur letzten Zeile
w	Nächstes Wort
W	Nächstes Wort getrennt durch Leerzeichen
b	Vorheriges Wort
B	Vorheriges Wort getrennt durch Leerzeichen
e	Wortende
E	Ende des Worts getrennt durch Leerzeichen

ge	Ende des vorheriges Wortes
gE	Ende des vorherigen Worts getrennt durch Leerzeichen
|	Gefolgt von einer Spaltennummer, um in die Spalte zu springen
%	Gefolgt von einem Element, um zu dem Element zu springen
`	Marker
`	Erstes Zeichen, das kein Leerzeichen ist, in der Zeile, in der sich der Marker befindet
[[	Vorherige öffnende eckige Klammer
]]	Nächste öffnende eckige Klammer
[]	Vorherige schließende eckige Klammer
][	Nächste schließende eckige Klammer
Strg+**I**	Geht zur nächsten Adresse
Strg+**O**	Geht zur vorherigen Adresse
H	Geht zur ersten Zeile auf dem Bildschirm
M	Geht zur mittleren Zeile auf dem Bildschirm
L	Geht zur letzten Zeile auf dem Bildschirm
%Prozentsatz	Geht zum angegebenen Prozentsatz des Dokuments
gk	Geht optisch eine Zeile aufwärts (bei dynamischem Zeilenumbruch)
gj	Geht optisch eine Zeile abwärts (bei dynamischem Zeilenumbruch)
Strg+**Pfeil links**	Verschiebt ein Wort nach links
Strg+**Pfeil rechts**	Verschiebt ein Wort nach rechts

9.1.6 Unterstützte Textobjekte

Diese können verwendet werden, um bestimmte Bereiche eines Dokuments auszuwählen.

iw	Inneres Wort: Wort inklusive Leerzeichen
aw	Ein Wort: Wort ohne Leerzeichen
i"	Vorherige Anführungszeichen (") bis nächste Anführungszeichen, inklusive der Anführungszeichen
a"	Vorherige Anführungszeichen (") bis nächste Anführungszeichen, ohne die Anführungszeichen
i'	Vorherige einfache Anführungszeichen (') bis nächste einfache Anführungszeichen, inklusive der Anführungszeichen
a'	Vorherige einfache Anführungszeichen (') bis nächste einfache Anführungszeichen, ohne die Anführungszeichen
i(	Vorherige öffnende Klammer [**(**] bis nächste schließende Klammer [**)**], inklusive der Klammern

a(	Vorherige öffnende Klammer [**(**] bis nächste schließende Klammer [**)**], ohne die Klammern
i[	Vorherige öffnende eckige Klammer (**[**) bis nächste schließende eckige Klammer (**]**), inklusive der Klammern
a[	Vorherige öffnende eckige Klammer (**[**) bis nächste schließende eckige Klammer (**]**), ohne die Klammern
i{	Vorherige öffnende geschweifte Klammer (**{**) bis nächste schließende geschweifte Klammer (**}**), inklusive der Klammern
a{	Vorherige öffnende geschweifte Klammer (**{**) bis nächste schließende geschweifte Klammer (**}**), ohne die Klammern
i<	Vorherige öffnende spitze Klammer (**<**) bis nächste schließende spitze Klammer (**>**), inklusive der Klammern
a<	Vorherige öffnende spitze Klammer (**<**) bis nächste schließende spitze Klammer (**>**), ohne die Klammern
i`	Vorheriges Backtick („rückwärts geneigtes Hochkomma“) (˜) bis nächstes Backtick, inklusive der Backticks
a`	Vorheriges Backtick („rückwärts geneigtes Hochkomma“) (˜) bis nächstes Backtick, ohne die Backticks

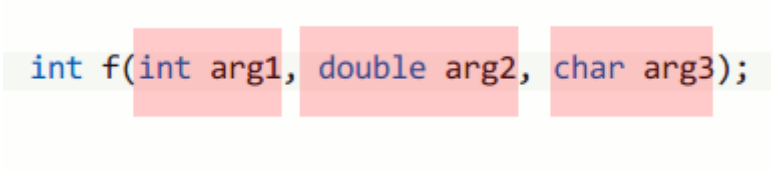
9.1.7 Unterstützte Befehle im Eingabemodus

Strg+D	Einrückung verringern
Strg+T	Einrücken
Strg+E	Von unten einfügen
Strg+Y	Wort löschen
Strg+W	Wort löschen
Strg+U	Zeile löschen
Strg+J	Neue Zeile
Strg+H	Löscht Zeichen rückwärts
Strg+Pos 1	Geht zum ersten Zeichen im Dokument
Strg+R n	Fügt den Inhalt des Registers `n` ein
Strg+O, *Befehl*	Wechselt für einen Befehl zum normalen Modus
Strg+A	Verringert die aktuell gewählte Zahl
Strg+X	Erhöht die aktuell gewählte Zahl

9.1.8 Das Komma-Textobjekt

Dieses Objekt fehlt in Vim. Das Komma-Text-Objekt vereinfacht das Ändern von Parameterlisten in C-ähnlichen Sprachen und anderen durch Komma getrennte Listen. Dies ist der Bereich zwi-

schen zwei Kommas oder einem Komma und einer Klammer. In der Demonstrationszeile sind die Bereiche, die dieses Text-Objekt umfassen kann, hervorgehoben.



Bereiche des Komma-Text-Objekts. Wenn sich der Cursor z. B. über `arg2` *befindet, bewirkt das Drücken von* **`ci`***(„inneres Komma ändern"), dass* `double arg2` *gelöscht wird und der Cursor im Eingabemodus zwischen die beiden Kommas platziert wird. Das ist ein sehr angenehme Art, Funktionsparameter zu ändern.*

9.1.9 Fehlende Funktionen

Wie bereits erwähnt, ist es nicht Ziel des VI-Modus in KatePart, die Funktionen von Vim zu 100 % zu unterstützen.

Anhang A

Reguläre Ausdrücke

Dieser Anhang enthält eine kurze, aber hoffentlich ausreichende Einführung in die Welt der *regulären Ausdrücke*. Es werden reguläre Ausdrücke in der Form dokumentiert, in der sie in KatePart anwendbar sind, die aber nicht kompatibel z. B. zu der in Perl oder in **grep** verwendeten Form ist.

A.1 Einleitung

Reguläre Ausdrücke stellen eine Möglichkeit zur Verfügung, vielleicht zu suchende Teile von Text in einer Form zu beschreiben, die von einer kleinen Software verstanden wird, sodass diese feststellen kann, ob die Beschreibung zutrifft und sogar Text zur späteren Verwendung speichern kann.

Ein Beispiel: Nehmen Sie an, Sie wollen eine Text nach Abschnitten durchsuchen, die mit einem der Namen „Henrik" oder „Pernille" beginnen, gefolgt von einer Form des Verbs „say".

Mit einer normalen Suche würden Sie anfangen, nach dem ersten Namen „Henrik" zu suchen, vielleicht gefolgt von „sa", also **`Henrik sa`**. Bei dieser Suche würden Sie alle Übereinstimmungen überspringen müssen, die nicht am Anfang eines Abschnittes stehen und die, hinter denen ein „sa" steht, aber kein „says", „said" und so weiter. Dann natürlich das Ganze von vorn für den nächsten Namen ...

Mit regulären Ausdrücken können Sie dies mit einer einzelnen Suche erreichen und das noch genauer.

Um dies zu erreichen, definieren reguläre Ausdrücke Regeln zum Ausdrücken von Details einer zu suchenden Zeichenfolge. Unser Beispiel, das wir wie folgt ausdrücken können: „Eine Zeile mit „Henrik" oder „Pernille" beginnend (eventuell nach bis zu 4 Leerzeichen oder Tabulatoren) gefolgt von einem Leerzeichen gefolgt von „sa" und dann entweder „ys" oder „id"" kann so als regulärer Ausdruck geschrieben werden:

```
^[ \t]{0,4}(Henrik|Pernille) sa(ys|id)
```

Das oben angegebene Beispiel zeigt alle vier Hauptkonzepte von regulären Ausdrücke, speziell:

- Muster
- Behauptungen
- Quantifiers
- Rückwärtsreferenzen

Das Hochzeichen (^) am Anfang des Ausdruckes kennzeichnet eine Behauptung, die nur dann wahr ist, wenn sich der folgende Text am Anfang einer Zeile befindet.

Die Zeichenfolgen `[ \t]` und `(Henrik|Pernille) sa(ys|id)` sind Muster. Das erste ist ein *Zeichen* das entweder auf ein Leerzeichen oder ein Tabulatorzeichen zutrifft, das andere enthält als erstes ein Untermuster, das entweder auf `Henrik` *oder* `Pernille` zutrifft, dann ein Muster, das exakt auf `sa` zutrifft und zum Schluss wieder ein Untermuster, das auf `ys`, *oder* `id` zutrifft.

Die Angabe `{0,4}` ist ein Quantifizierer, der sagt: „von 0 bis 4 mal das vorher spezifizierte".

Weil Software für reguläre Ausdrücke, die das Konzept von *Referenzen* den gesamten zutreffenden Teil des Textes wie auch in Klammern eingeschlossene Untermuster sichert, können Sie diese gefundenen Textstellen (diese sind nach dem Suchen mit einem regulären Ausdruck in einem Textdokument in einem Editor meistens ausgewählt) oder den gefundenen Namen oder den letzten Teil des Verbs weiterverwenden, die *Referenzen* ermöglichen den Zugriff auf diese.

Zusammengefasst: die regulären Ausdrücke treffen zu, wo wir wollten, und nur dort.

Die folgenden Abschnitte beschreiben im einzelnen, wie Muster, Zeichenklassen, Behauptungen, Quantifizierer und Rückwärtsreferenzen benutzt werden und der letzte Abschnitt gibt einige nützliche Beispiele.

A.2 Muster

Muster bestehen aus Zeichenfolgen und Zeichenklassen. Muster können Untermuster enthalten, diese sind in Klammern eingeschlossene Muster.

A.2.1 Steuerzeichen

In Mustern und in Zeichenklassen haben einige Zeichen spezielle Bedeutungen. Um diese Steuerzeichen zu finden, müssen sie als solche markiert werden.

Dies geschieht durch das Voranstellen eines Rückwärtsschrägstriches (\) vor das Zeichen.

Die Software ignoriert die Kennzeichnung als Steuerzeichen von Zeichen, die in dem betrachteten Zusammenhang keine Steuerzeichen sind. So ist z. B. die Angabe von (**`\j`**), also ein „j" als Steuerzeichen markiert, kein Problem. Wenn Sie Zweifel haben, ob ein Zeichen eine spezielle Bedeutung hat, können Sie dies ohne Bedenken als Steuerzeichen markieren.

Selbstverständlich können Sie auch den Rückwärtsschrägstrich als Steuerzeichen markieren, dies geschieht durch **`\\`**.

A.2.2 Zeichenklassen und Abkürzungen

Eine *Zeichenklasse* ist ein Ausdruck, der auf einen bestimmten Satz von Zeichen zutrifft. Zeichenklassen werden in regulären Ausdrücken durch Setzen der zugelassenen Zeichen in eckige Klammern `[]` oder durch Nutzen einer der im Folgenden beschriebenen abgekürzten Klassen definiert.

Einfache Zeichenklassen enthalten nur ein oder mehrere Zeichen, z. B. **`[abc]`** (zutreffend auf einen der Buchstaben „a", „b" oder „c") oder **`[0123456789]`** (zutreffend auf eine Zahl).

Da Buchstaben und Zahlen eine festgelegte Reihenfolge haben, können diese durch Angabe des Bereiches :abgekürzt werden: **`[a-c]`** entspricht **`[abc]`** und **`[0-9]`** entspricht **`[0123456789]`**. Diese Angaben können auch kombiniert werden, zum Beispiel trifft **`[a-fynot1-38]`** auf die folgenden Zeichen zu: „a", „b", „c", „d", „e", „f", „y", „n", „o", „t", „1", „2", „3" oder „8".

Da Großbuchstaben von Kleinbuchstaben unterschieden werden, müssen Sie zur Angabe von „a" oder „b" ohne Unterscheidung von Groß- und Kleinschreibung **`[aAbB]`** angeben.

Die Erzeugung von „negativen" Klassen, die auf „alles außer" zutreffen, erfolgt durch das Hoch-Zeichen (^) am Anfang der Klassendefinition:

[^abc] trifft auf alle Zeichen *außer* „a", „b" oder „c" zu.

Zusätzlich zu den druckbaren Zeichen sind noch einige Abkürzungen definiert, um die Verwendung ein wenig einfacher zu machen:

\a
: Trifft auf das ASCII-Beep-Zeichen zu (BEL, 0x07).

\f
: Trifft auf das ASCIISeitenvorschub-Zeichen zu (FF, 0x0C).

\n
: Trifft auf das ASCII-Zeilenvorschub-Zeichen zu (LF, 0x0A, Unix newline).

\r
: Trifft auf das ASCII-Wagenrücklauf-Zeichen zu (CR, 0x0D).

\t
: Trifft auf das ASCII-Zeichen Horizontaltabulator zu (HT, 0x09).

\v
: Trifft auf das ASCII-Zeichen Vertikaltabulator zu (VT, 0x0B).

\xhhhh
: Dieser Ausdruck trifft auf das Unicodezeichen mit dem Code mit der Hexadezimalzahl hhhh (zwischen 0x0000 und 0xFFFF) zu. \0ooo (d. h., \zero ooo) trifft auf das ASCII-/Latin-1-Zeichen mit dem Code mit der Oktalzahl ooo (zwischen 0 und 0377) zu.

. (Punkt)
: Trifft auf jedes Zeichen einschließlich Zeilenvorschub zu.

\d
: Trifft auf eine Ziffer zu. Entspricht `[0-9]`.

\D
: Trifft auf ein Zeichen, das keine Ziffer ist, zu. Entspricht `[^0-9]` oder `[^\d]`.

\s
: Trifft auf ein Zeichen, das einen Zwischenraum angibt, zu. Praktisch entspricht dies `[ \t\n\r]`.

\S
: Trifft auf ein Zeichen, das keinen Zwischenraum angibt, zu. Praktisch entspricht dies `[^ \t\n\r]` oder `[^\s]`.

\w
: Trifft auf jedes „druckbares Zeichen" zu - in diesem Fall Buchstaben, Ziffern oder Unterstrich. Entspricht `[a-zA-Z0-9]`

\W
: Trifft auf alle nicht druckbaren Zeichen außer Buchstaben, Ziffern und Unterstrich zu. Entspricht `[^a-zA-Z0-9]` oder `[^\w]`

Die *POSIX-Notation von Klassen*, **[:<class name>:]** wird auch unterstützt. Zum Beispiel entspricht **[:digit:] \d** und **[:space:] \s**. Die vollständige Liste der POSIX-Zeichenklassen finden Sie hier.

Die abgekürzten Klassen können in selbstdefinierte Klassen eingefügt werden, z. B. kann für die Klasse „druckbares Zeichen, Leerzeichen oder Punkt" der Ausdruck **[\w \.]** verwendet werden.

A.2.2.1 Zeichen mit speziellen Bedeutungen (Steuerzeichen) innerhalb von Zeichenklassen

Die folgenden Zeichen haben spezielle Bedeutungen innerhalb des Ausdrucks in eckigen Klammern „[]", diese müssen als Steuerzeichen gekennzeichnet werden, damit sie als Zeichen in die Klasse einbezogen werden:

]

Beendet die Definition der Zeichenklasse. Dieses Zeichen braucht nicht als Steuerzeichen gekennzeichnet werden, wenn es das erste Zeichen in einer Zeichenklassendefinition (nach dem Zeichen „[" oder „^") ist.

^ (Hoch-Zeichen)

Bezeichnet eine negative Klasse, wenn es das erste Zeichen in einer Zeichenklassendefinition ist. Wenn es als druckbares Zeichen behandelt werden soll, muss es als Steuerzeichen gekennzeichnet werden, wenn es das erste Zeichen in einer Zeichenklassendefinition ist.

– (Bindestrich)

Kennzeichnet einen logischen Bereich. Wenn es als Zeichen behandelt werden soll, muss es als Steuerzeichen gekennzeichnet werden.

\ (Rückwärtsschrägstrich)

Das Zeichen zum Kennzeichnen eines Steuerzeichens. Dieses Zeichen muss immer als Steuerzeichen gekennzeichnet werden, wenn es als druckbares Zeichen behandelt werden soll.

A.2.3 Alternativen: trifft zu wenn „eins von"

Wenn ein erkanntes Muster von mehreren Mustern als zutreffend erkannt werden soll, dann müssen Sie diese Muster durch einen senkrechten Strich | getrennt angeben.

Der Ausdruck **`John|Harry`** wird z. B. als zutreffend erkannt, wenn entweder „John" oder „Harry" gefunden wird.

A.2.4 Untermuster

Untermuster sind in Klammern eingeschlossene Muster, die in regulären Ausdrücken viele Verwendungen haben.

A.2.4.1 Angabe von Alternativen

Sie können Untermuster verwenden, um Gruppen von Alternativen in einem Muster anzugeben. Die Alternativen werden durch den senkrechten Strich | getrennt.

Um eines der Worte „int", „float" oder „double" zu erkennen, geben Sie **`int|float|double`** an. Wenn eines der Worte nur dann erkannt werden soll, wenn nach dem Wort Zwischenraum und dann Buchstaben folgen, dann verwenden Sie den folgenden Ausdruck mit den Worten im Untermuster: **`(int|float|double)\s+\w+`**.

A.2.4.2 Speichern von gefundenem Text (Rückwärtsreferenzen)

Wenn Sie einen Rückverweis verwenden möchten, benutzen Sie ein Untermuster **`(PATTERN)`**, um den gewünschten Teil des Musters zu merken. Um zu verhindern, dass das Untermuster gemerkt wird, verwenden Sie eine nicht speichernde Gruppe **`(?:PATTERN)`**.

Wenn Sie z. B. das zweifache Auftreten des selben Wortes getrennt durch ein Komma und eventuell Zwischenraum finden wollen, dann würden Sie **`(\w+),\s*\1`** verwenden. Das Untermuster

`\w+` findet ein Stück aus druckbaren Zeichen. Der gesamte Ausdruck trifft zu, wenn diese von einem Komma und keinem oder mehreren Zwischenraumzeichen und dann von einem gleichen Stück von Zeichen gefolgt werden. (Der Ausdruck `\1` verweist auf das *erste in Klammern angegebene Untermuster*.)

ANMERKUNG

Um Mehrdeutigkeiten bei Ausdrücken wie **`\1`** und einigen nachfolgenden Ziffern wie z. B. **`\12`** - 12tes Untermuster oder nur das erste Untermuster mit **`2`** zu vermeiden, wird die Syntax **`\{12}`** für Untermuster aus mehreren Ziffern verwendet.
Beispiele:

- **`\{12}1`** bedeutet „Untermuster 12 benutzen”
- **`\123`** bedeutet „1 ist das Suchmuster und 23 normaler Text”

A.2.4.3 Vorwärtsgerichtete Behauptungen

Eine vorwärtsgerichtete Behauptung ist ein Untermuster, das mit `?=` oder `?!` anfängt.

Der Ausdruck **`Bill(?! Gates)`** besagt, dass „Bill” gefunden wird, aber nur wenn nicht von „ Gates” gefolgt. Dies findet „Bill Clinton” oder „Billy the kid”, aber ignoriert stillschweigend die andere Übereinstimmung.

Untermuster, die für Behauptungen benutzt werden, werden nicht gespeichert.

Sehen Sie auch unter Behauptungen nach.

A.2.4.4 Rückwärtsgerichtete Behauptungen

Eine rückwärtsgerichtete Behauptung ist ein Untermuster, das entweder mit `?<=` oder mit `?<!` anfängt.

Eine Rückwärtsreferenz hat die gleiche Wirkung wie eine Vorwärtsreferenz, funktioniert aber rückwärts. Um zum Beispiel die Zeichenfolge „fruit” zu finden, aber nur, wenn nicht „grape” vorangestellt ist, könnten Sie diesen Ausdruck verwenden: **`(?<!grape)fruit`**.

Untermuster, die für Behauptungen benutzt werden, werden nicht gespeichert.

Sehen Sie auch unter Behauptungen nach.

A.2.5 Zeichen mit speziellen Bedeutungen (Steuerzeichen) innerhalb von Mustern

Die folgenden Zeichen haben spezielle Bedeutungen innerhalb eines Musters, diese müssen als Steuerzeichen gekennzeichnet werden, damit sie als Zeichen behandelt werden:

\ (Rückwärtsschrägstrich)
: Das Escape-Zeichen.

^ (Hoch-Zeichen)
: Kennzeichnet den Anfang der Zeichenfolge.

$
: Kennzeichnet das Ende der Zeichenfolge.

() (linke und rechte Klammer)
: Kennzeichnet Untermuster.

{} (linke und rechte geschweifte Klammer)
: Kennzeichnet numerische Quantifizierer.

[] (linke und rechte eckige Klammer)
: Kennzeichnet Zeichenklassen.

| (senkrechter Strich)
: Logisches ODER. Trennt Alternativen.

+ (Pluszeichen)
: Quantifizierer, steht für eins oder mehrere.

*** (Stern)**
: Quantifizierer, steht für kein oder mehrere.

? (Fragezeichen)
: Ein optionales Zeichen. Kann als Quantifizierer; 0- oder 1-mal gedeutet werden.

A.3 Quantifizierer

Quantifizierer gestatten dem regulären Ausdruck die Angabe einer Anzahl von entweder Zeichen, Zeichenklassen oder Untermustern.

Quantifizierer werden in geschweifte Klammern (`{` und `}`) eingeschlossen und haben die Form `{[minimale Anzahl][,[maximale Anzahl]]}`

Die Benutzung ist am besten an Beispielen erklärt:

{1}
: Genau einmaliges Auftreten

{0,1}
: Kein oder einmaliges Auftreten

{,1}
: Kein oder einmaliges Auftreten (Kurzform)

{5,10}
: Mindestens 5- bis maximal 10-maliges Auftreten

{5,}
: Mindestens 5-maliges Auftreten.

Zusätzlich gibt es einige Abkürzungen:

*** (Stern)**
: entspricht `{0,}` findet jede Anzahl des Auftretens.

+ (Pluszeichen)
: entspricht `{1,}` findet mindestens einmaliges Auftreten.

? (Fragezeichen)
: entspricht `{0,1}` findet kein oder einmaliges Auftreten.

A.3.1 Gier

Wenn Quantifizierer ohne Maximum verwendet werden, dann findet der reguläre Ausdruck so viel wie möglich vom Suchtext, dieses Verhalten wird auch als *gierig* bezeichnet.

Moderne Software für reguläre Ausdrücke stellt die Möglichkeit bereit, das „gierige Verhalten auszuschalten", aber in einer grafischen Umgebung ist es das Interface, das Ihnen Zugriff auf diese Möglichkeit bereitstellen muss. Ein Dialogfenster zum Suchen kann z. B. eine Option mit dem Namen „Minimales Finden" bereitstellen, es sollte auch anzeigen, ob „gieriges Verhalten" Standard ist.

A.3.2 In Beispielen

Hier sind einige Beispiele der Verwendung von Quantifizierern:

`^\d{4,5}\s`
: Trifft auf die Zahlen in „1234 go" und „12345 now" zu, aber nicht die in für „567 eleven" oder „223459 somewhere".

`\s+`
: Trifft auf ein oder mehrere Zwischenraumzeichen zu.

`(bla){1,}`
: Trifft zu für alle in „blablabla" und das „bla" in „blackbird" oder „tabla".

`/?>`
: Trifft für das „/>" in „<closeditem/>" sowie auch für das „>" in „<openitem>" zu.

A.4 Behauptungen

Behauptungen erweitern den regulären Ausdruck so, dass er nur unter bestimmten Bedingungen zutrifft.

Eine Behauptung braucht kein Zeichen um zuzutreffen, diese ermittelt vielmehr die Umgebung einer eventuellen Übereinstimmung bevor dieser bestätigt wird. Die Behauptung *Wortgrenze* z. B. versucht nicht, ein nichtdruckbares Zeichen neben einem druckbaren Zeichen zu finden, sondern stellt fest, dass dort KEIN druckbares Zeichen ist. Das heißt, dass dieses z. B. auch am Ende einer Zeichenfolge zutrifft.

Einige Behauptungen haben ein Muster das gefunden werden muss, aber der zutreffende Teil des Suchtextes dieses Musters wird nicht Teil des Ergebnisses des gesamten regulären Ausdrucks.

Reguläre Ausdrücke wie hier beschrieben unterstützen die folgenden Behauptungen:

`^` (Hochzeichen: Anfang der Zeichenfolge)
: Trifft auf den Anfang des zu suchenden Textes zu.

  Der Ausdruck **`^Peter`** trifft auf „Peter" im Text „Peter, hey!" zu, aber nicht auf „Hey, Peter!".

`$` (Ende der Zeichenfolge)
: Trifft auf das Ende des Suchtextes zu.

  Der Ausdruck **`you\?$`** trifft auf das letzte „you" in „You didn‚t do that, did you?" zu, aber nirgendwo in „You didn't do that, right?".

\b (Wortgrenze)

Trifft zu, wenn ein druckbares Zeichen auf der einen Seite aber keines auf der anderen Seite ist.

Dieser Ausdruck dient zum Finden von Wortenden, wenn nach beiden Enden gesucht wird, zum Finden des ganzen (einzelnstehenden) Wortes. Der Ausdruck **\bin\b** trifft auf das einzelnstehende „in" in „He came in through the window" zu, aber nicht auf das „in" in „window".

\B (keine Wortgrenze)

Trifft immer dort zu, wo „\b" nicht zutrifft.

Dieser Ausdruck dient zum Finden von Text innerhalb von Worten. Der Ausdruck **\Bin\B** trifft z. B. auf das „in" im Wort „window" im Text „He came in through the window" zu, aber nicht auf „integer" oder „I'm in love".

(?=PATTERN) (Positive Vorwärtsreferenz)

Eine Vorwärtsreferenz prüft den Text, der dem eventuell zutreffenden Teil des Textes folgt. Die Vorwärtsreferenz verhindert, dass der Text zutrifft, wenn der nachfolgende Text nicht auf das *MUSTER* der Behauptung zutrifft. Wenn die Behauptung zutrifft, wird der Text, der auf diese zutrifft, allerdings nicht Bestandteil des Ergebnisses.

Der Ausdruck **handy(?=\w)** trifft auf „handy" in „handyman" zu, aber nicht auf das in „That came in handy!"

(?!PATTERN) (Negative Vorwärtsreferenz)

Eine negative Vorwärtsreferenz verhindert, dass der Text zutrifft, wenn der nachfolgende Text auf das *MUSTER* zutrifft.

Der Ausdruck **const \w+\b(?!\s*&)** trifft auf „const char" im Text „const char* foo", aber nicht „const QString" in „const QString& bar" weil das „&" auf die negative Vorwärtsreferenz zutrifft.

(?=PATTERN) (Positive Rückwärtsreferenz)

Eine Rückwärtsreferenz hat die gleiche Wirkung wie eine Vorwärtsreferenz, funktioniert aber rückwärts. Eine Rückwärtsreferenz sucht den Teil der Zeichenfolge vor einer mögliche Übereinstimmung. Die positive Rückwärtsreferenz passt nur dann auf eine Zeichenfolge, wenn *PATTERN* der Behauptung vorangestellt ist, aber der damit übereinstimmende Text wird nicht in das Ergebnis aufgenommen.

Der Ausdruck **(?<=cup)cake** passt auf „cake", wenn „cup" vorangestellt ist wie z. B. in „cupcake" aber nicht in „cheesecake" oder in nur „cake".

(?!PATTERN) (Negative Rückwärtsreferenz)

Eine negative Rückwärtsreferenz verhindert, dass der Text zutrifft, wenn der vorherige Text auf das *MUSTER* zutrifft.

Der Ausdruck **(?<![\w\.])[0-9]+** passt auf „123" in der Zeichenfolgen „=123" und „-123", aber nicht auf „123" in „.123" oder „word123".

(PATTERN) (Gruppierung)

Das Untermuster innerhalb der Klammern wird gespeichert und gemerkt, so dass es in Rückwärtsreferenzen verwendet werden kann. Der Ausdruck **("+)[^"]*\1** passt zum Beispiel auf ˝˝˝˝**text**˝˝˝˝ und ˝**text**˝.

Weitere Informationen im Abschnitt Speichern von gefundenem Text (Rückwärtsreferenzen).

(?:PATTERN) (Nicht speichernde Gruppe)

Das Untermuster in den Klammern wird nicht gespeichert und gemerkt. Sie sollten immer nicht speichernde Gruppen zu verwenden, wenn die Speicherungen nicht verwendet werden.

Anhang B

Index

E
Ersetzen, sed-Stil
    Suchen, sed-Stil, 36

K
Kommentar, 32
Kommentar entfernen, 32