编写语法突出显示文件

2005年3月24日星期四 | 克里斯托夫·卡尔曼

注意：请参阅Kate手册，了解如何编写语法突出显示文件的最新版本。

提示：如果你想写一个语法高亮文件，XML完成插件可能会有很大帮助。

本节概述了 KDE4 中的突出显示定义 XML 格式。基于一个小示例，它将描述主要组件及其含义和用法。下一节将详细介绍突出显示检测规则。

凯特突出显示定义文件的主要部分

正式定义，即 DTD 存储在文件中该文件应安装在系统上的文件夹中。未设置 ifis 使用查找文件夹。
`language.dtd` `$KDEDIR/share/apps/katepart/syntax` `$KDEDIR` `kde4-config --prefix`

一个例子

突出显示文件包含一个标头，用于设置 XML 版本和文档类型：

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE language SYSTEM "language.dtd">
```

定义文件的根是元素**语言**。可用属性包括：

必需属性：

- **名称**设置语言的名称。它显示在菜单和对话框中。
- **部分**指定类别。
- 扩展名定义文件**扩展**名，如“*.cpp;*。h"

可选属性：

- **Mimetype**关联文件 Mime 类型基于。
- 版本指定定义文件的当前**版本**。
- **kateversion**指定 Kate 的最新受支持版本。
- 区分大小写定义关键字是否**区分大小写**。 注意：尚未实现。
- 如果另一个突出显示定义文件使用相同的扩展名，则**优先级**是必需的。优先级越高者将获胜。
- 作者包含**作者**的姓名和他的电子邮件地址。
- 许可证包含**许可证**，通常是LGPL，Artistic，GPL等。指定许可证很重要，因为 kate 团队需要一些法律支持来分发文件。
- **隐藏**定义，名字是否应该出现在凯特的菜单中。

所以下一行可能看起来像这样：

```
<language name="C++" version="1.00" kateversion="2.4" section="Sources" extensions="*.cpp;*.h" >
```

接下来是**突出显示**元素，其中包含可选元素**列表**以及所需的元素**上下文**和itemData。

列表元素包含关键字**列表**。在本例中，关键字是*class*和*const*。您可以根据需要添加任意数量的列表。**上下文**元素包含所有上下文。默认情况下，第一个上下文是突出显示的开始。上下文“*普通文本*”中有两个规则，它们将关键字列表与名称*somename*匹配，以及一个检测引号并将上下文切换为*字符串*的规则。要了解有关规则的更多信息，请阅读下一章。
第三部分是**itemDatas**元素。它包含上下文和规则所需的所有颜色和字体样式。
在此示例中，使用**项目数据***普通文本*，*字符串*和*关键字*。

```
<highlighting> <list name="somename"> <item> class </item> <item> const </item> </list> <contexts> <cc
```

突出显示定义的最后一部分是可选的**常规**部分。它可能包含有关关键字，代码折叠，注释和缩进的信息。

注释部分定义引入单行**注释**的字符串。您还可以使用带有附加属性*末尾*的多行来定义*多行*注释。如果用户按下相应的*注释/取消注释*快捷方式，则使用此方法。
关键字部分定义**关键字**列表是否区分大小写。其他属性将在后面解释。

```
<general> <comments> <comment name="singleLine" start="#"/> </comments> <keywords casesensitive="1"/>
```

各部分详细介绍

这部分将描述上下文，itemDatas，关键字，注释，
代码折叠和缩进的所有可用属性。

元素上下文属于组**上下文**。上下文本身定义了特定于上下文的规则，例如如果突出显示系统到达行尾时会发生什么。可用属性包括：

- **命名**上下文名称/标识符。规则将使用此名称指定在规则匹配时要切换到的上下文。
- **属性**在当前上下文中没有匹配规则时使用的默认项目数据。
- **lineEndContext**定义突出显示系统在到达行尾时切换到的上下文。这可能是另一个上下文的名称，**#stay**不切换上下文（即什么都不做），或者**#pop**这将导致离开此上下文。例如，可以使用#pop#pop**#pop来**弹出三次。
- **lineBeginContext**定义遇到一行开头时的上下文。默认值：#stay。
- fallthrough定义如果没有匹配的规则，突出显示系统是否切换到**fallthroughContext**中指定的上下文。默认值：*假*。
- **fallthroughContext**指定没有规则匹配时的下一个上下文。
- **动态**如果*为 true*，则上下文会记住动态规则保存的字符串/占位符。例如，*对于HERE*文档，这是必需的。默认值：*假*。

元素**itemData**位于组**itemDatas** 中。它定义了字体样式和颜色。因此，可以定义自己的样式和颜色，但是我们建议尽可能坚持使用默认样式，以便用户始终看到不同语言中使用的相同颜色。但是，有时没有其他方法，有必要更改颜色和字体属性。属性名称和 defStyleNum 是必需的，另一个是可选的。可用属性包括：

- **名称**设置项数据的名称。上下文和规则将在其属性中使用此名称。
- **属性**来引用项数据。
- **defStyleNum**定义要使用的默认样式。后面将详细介绍可用的默认样式。
- **颜色**定义颜色。有效格式为“#rrggbb”或“#rgb”。
- **selColor**定义选区颜色。
- **斜体**如果*为 true*，则文本将为*斜体*。
- **粗体**如果*为 true*，则文本将为**粗体**。
- **下划线**如果*为 true*，则文本将带有下划线。

- **删除线**如果*为 true*，则文本将被~~划出~~。
- **拼写检查**如果为 true，则将对文本进行拼写检查，否则在拼写检查期间将忽略该文本。

组常规中的元素关键字定义**关键字**属性。可用属性包括：

- **区分大小写**可以是*真的*，也可以*是假*的。如果*为 true*，则所有关键字都匹配区分大小写。默认值：*真*。
- **弱 Deliminator***是不充当单词*分隔符的字符列表。例如，点 （.） 是单词分隔符。假设**列表中的**关键字包含一个点，则仅当您将该点指定为弱分隔符时，它才会匹配。
- **另外，**Deliminator定义了其他分隔符。
- wordWrapDeliminator定义字符，之后可能会出现换行。默认分隔符和自动换行分隔符是字符。 **（）：！+，-<=>%&/;?** []^{|}~_*、**空格** （' '） 和**制表符** （\t）。

组注释中的元素**注释**定义用于
and 的注释属性。
可用属性包括： `Tools > Comment` `Tools > Uncomment`

- **名称**为*单行*或*多行*。
- 如果选择*单行*，则可选属性**位置**可用。此属性的默认值是在列 0 中插入单行注释字符串。如果您希望它出现在空格之后，则必须将其设置为*“空格后”*，例如：。 `position="afterwhitespace"`
- 如果选择*多线*，则属性*端*和*区域*是必需的。
- start定义用于开始注释的字符串。C++这是。 `/*`
- end定义用于关闭注释的字符串。C++这是。 `*/`
- **区域**应该是可折叠多行注释的名称。假设您有*开始区域=“注释”...endRegion=“Comment”*在你的规则中，你应该使用*region=“Comment”*。即使您没有选择多行注释的所有文本，这种取消注释的方式也有效。光标只能位于多行注释中。

常规组中的元素折叠定义代码**折叠**属性。可用属性包括：

- **缩进敏感**如果*为 true*，则代码折叠标记将添加基于缩进，就像在脚本语言Python 中一样。通常你不需要设置它，因为它默认为*false*。

常规组中的元素缩进定义了将使用哪个缩进器，但是我们强烈建议省略此元素，因为**缩**进器通常通过定义文件类型或向文本文件添加模式行来设置。但是，如果您指定缩进器，则会强制用户使用特定的缩进，他可能根本不喜欢。
可用属性包括：

- **模式**是压头的名称。目前可用的压头有：*none*，*normal*，*cstyle*，*haskell*，*lilypond*，*lisp*，*python*，*ruby*和*xml*。

可用的默认样式

默认样式是预定义的字体和颜色样式。为方便起见，Kate 详细介绍了几种默认样式：

- dsNormal，用于普通文本。
- ds关键字，用于关键字。
- dsDataType，用于数据类型。
- dsDecVal，用于十进制值。
- dsBaseN，用于基数不是 10 的值。
- dsFloat，用于浮点值。
- dsChar，用于字符。
- dsString，用于字符串。
- dsComment，用于注释。

- dsOthers，用于“其他”事物。
- dsAlert，用于警告消息。
- dsFunction，用于函数调用。
- dsRegionMarker，用于区域标记。
- dsError，用于错误突出显示和错误语法。

突出显示检测规则

本节介绍语法检测规则。

每个规则可以匹配测试所依据的字符串开头的零个或多个字符。如果规则匹配，则为匹配的字符分配规则定义的样式或*属性*，并且规则可能会要求切换当前上下文。

规则如下所示：

*该属性*按名称标识要用于匹配字符的样式，*上下文*标识要从此处使用的上下文。

*上下文*可以通过以下方式标识：

- *标识符*，它是另一个上下文的名称/标识符。
- 指示引擎保留在当前上下文 （**#stay**） 或弹出到字符串中使用的先前上下文 （**#pop**） 的*顺序*。要返回更多步骤，可以重复#pop关键字：**#pop#pop#pop**

某些规则可以具有子规则，然后仅在父规则匹配时才评估这些*子规则*。整个匹配的字符串将被赋予父规则定义的属性。具有子规则的规则如下所示：

```
<RuleName (attributes)> <ChildRuleName (attributes) /> ... </RuleName>
```

特定于规则的属性各不相同，将在以下各节中介绍。

常见属性

所有规则都具有以下共同属性，并且只要出现**（通用属性）**即可使用。以下所有属性都是可选的。

- **属性**映射到定义的*项数据*。默认值：*目标上下文*中的属性
- 上下文指定突出显示系统在规则匹配时切换到的**上下文**。默认值：*#stay*
- beginRegion启动一个代码折叠块。默认值：未设置。
- endRegion关闭代码折叠块。默认值：未设置。
- **展望**，如果*为 true*，则突出显示系统将不会处理匹配长度。默认值：*假*。
- firstNonSpace，如果*为 true*，则规则仅在字符串是行中的第一个非空格时才匹配。默认值：*假*。
- 列定义**列**。仅当当前列与给定列匹配时，规则才匹配。默认值：未设置。

动态规则

某些规则允许默认为*false* 的布尔类型的可选属性**动态**。如果 dynamic 为*true*，则规则可以使用占位符来表示与其**字符串**或char属性中切换到当前上下文的*正则表达式*规则匹配的文本。在**字符串**中，占位符 **%N**（其中 N 是数字）将替换为调用正则表达式中的相应捕获**N**。在**char**中，placeholer 必须是数字 N，它将替换为调用正则表达式中相应捕获**N**的第一个字符。只要规则允许此属性，它就会包含 *a（动态）*。

- **动态**可以是*真的*，也可以是*假*的。默认值：*假*。

规则详解

侦查

检测单个特定字符。例如，通常用于查找带引号的字符串的结尾。

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

char属性定义要匹配的字符。

侦探2字符

按定义的顺序检测两个特定字符。

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

char属性定义要匹配的第一个字符，**char1**定义第二个字符。

AnyChar

检测一组指定字符中的一个字符。

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

字符串属性定义字符集。

字符串检测

检测确切的字符串。

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

字符串属性定义要匹配的**字符串**。**不敏感**属性默认为*false*，并传递给字符串比较函数。如果值为*true*，则使用不敏感的比较。

WordDetect （KDE >= 4.5， Kate >= 3.5）

检测确切的字符串，但还需要单词边界，如点 （.） 或单词开头和结尾的空格。您可以将*b*|*b*视为*正则表达式。

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

字符串属性定义要匹配的**字符串**。**不敏感**属性默认为*false*，并传递给字符串比较函数。如果值为*true*，则使用不敏感的比较。

正则表达式

与正则表达式匹配。

```
<RegExpr String="(string)" [insensitive="true|false"] [minimal="true|false"] (common attributes) (dynam
```

- **属性**定义正则表达式。
- **不敏感**默认为*false*，并传递给正则表达式引擎。
- **最小**默认值为*false*，并传递给正则表达式引擎。

由于规则始终与当前字符串的开头匹配，因此以插入符号 （^） 开头的正则表达式指示规则应仅与行首匹配。

关键词

从指定列表中检测关键字。

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

String属性按名称标识关键字列表。必须存在具有该名称的列表。

国际

检测整数。

```
<Int (common attributes) (dynamic) />
```

此规则没有特定属性。子规则通常用于检测数字后面的**L**和**U**组合，指示程序代码中的整数类型。实际上，所有规则都允许作为子规则，但是，DTD 只允许子规则**StringDetect**。
下面的示例匹配后跟字符“L”的整数。

```
<Int attribute="Decimal" context="#stay" > <StringDetect attribute="Decimal" context="#stay" String="L"
```

浮

检测浮点数。

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

此规则没有特定属性。**允许将 AnyChar**作为子规则，通常用于检测组合，请参阅规则**Int**作为参考。

HlCOct

检测八进制点数表示形式。

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

此规则没有特定属性。

赫克斯特

检测十六进制数表示形式。

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

此规则没有特定属性。

HlCtringChar

检测转义字符。

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

此规则没有特定属性。

它匹配程序代码中常用字符的文字表示形式，例如**\n**（换行符）或**\t**（制表符）。

如果以下字符跟在反斜杠 （***） 后，它们将匹配*: **abefnrtv"'? ***。此外，转义的十六进制数（例如**\xf)** 和转义的八进制数（例如**\033**）将匹配。

HlCChar

检测 C 字符。

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

此规则没有特定属性。

它匹配括在勾号中的 C 字符（例如：“**c**”）。所以在刻度中可以是一个简单的字符，也可能是一个转义的字符。有关匹配的转义字符序列，请参阅 HlCStringChar。

测距检测

检测具有定义的开始和结束字符的字符串。

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

char定义以范围开头的字符，**char1**定义结束范围的字符。可用于检测例如带引号的小字符串等，但请注意，由于突出显示引擎一次只处理一行，因此不会找到跨越换行符的字符串。

行继续

匹配反斜杠 ('’）在一行的末尾。

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

此规则没有特定属性。
如果最后一个字符是反斜杠 ('****')，则此规则对于切换行尾的上下文很有用。例如，在 C/C++ 中需要这样做才能继续宏或字符串。

包含规则

包含来自其他上下文或语言/文件的规则。

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

上下文属性定义要包含的**上下文**。
如果是简单字符串，它将所有定义的规则包含在当前上下文中，例如：

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

如果字符串以 **##** 开头，则突出显示系统将查找具有给定名称的另一种语言定义，例如：

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

如果**includeAttrib**属性为*true*，请将目标属性更改为源之一。例如，如果与包含的上下文匹配的文本与主机上下文的突出显示不同，则需要这样做才能使注释工作。

检测空间

检测空格。

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

此规则没有特定属性。
如果您知道前面可以有多个空格，例如缩进行的开头，请使用此规则。此规则将一次跳过所有空格，而不是测试多个规则并由于不匹配而一次跳过一个规则。

检测标识符

检测标识符字符串（作为正则表达式：[a-zA-Z_][a-zA-Z0-9_]*）。

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

此规则没有特定属性。
使用此规则一次跳过一串单词字符，而不是使用多个规则进行测试，并且由于不匹配而一次跳过一个规则。

提示和技巧

一旦您了解了上下文切换的工作原理，就很容易编写突出显示定义。尽管您应该仔细检查在什么情况下选择什么规则。正则表达式非常强大，但与其他规则相比，它们很慢。因此，您可以考虑以下提示。

- 如果只匹配两个字符，请使用**Detect2Chars**而不是**StringDetect**。这同样适用于**DetectChar**。
- 正则表达式易于使用，但通常还有另一种更快的方法可以实现相同的结果。假设您只想匹配字符 **#**，如果它是行中的第一个字符。基于正则表达式的解决方案如下所示：

  您可以使用以下方法更快地实现相同的目标：

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

  如果要匹配正则表达式“^#”，您仍然可以将**DetectChar**与属性**列=“0”**一起使用。属性**列**基于字符计数，因此制表器仍然只有一个字符。

  - 您可以在不处理字符的情况下切换上下文。假设您希望在遇到字符串**/* 时切换上下文，但需要在下一个上下文中处理该字符串。以下规则将匹配，并且** lookAhead属性将导致荧光笔为下一个上下文保留匹配的字符串。
  - 如果您知道出现许多空格，请使用**检测空间**。
  - 使用**DetectIdentifier**而不是正则表达式 '**[a-zA-Z_]\w***'。

- 尽可能使用默认样式。这样，用户将找到一个熟悉的环境。
- 查看其他 XML 文件，了解其他人如何实现棘手的规则。
- 您可以使用命令 `xmllint --dtdvalid language.dtd mySyntax.xml` 来验证每个 XML 文件。
- 如果您经常重复复杂的正则表达式，则可以使用*ENTITIES*。示例：]>

  现在您可以使用*&myref;*而不是正则表达式。

谢谢

资源

社区维基

用户库维基

支持

下载 KDE 软件

行为准则

隐私策略

应用程序隐私政策

目的地

凯德商店

KDE e.V.

KDE Free Qt 基金会

KDE 年表

KDE 宣言

国际网站