SignText – 标志文本

v1.3

1.简介

SignText 标志文本(.st)是基于 **MT 管理器**的 MT 语法文件(.mtsx)开发的一类语法，只具有用于文本编辑器的语法高亮功能，并不具有其他功能。以下规定的语法仅为格式规范与语法高亮正确，并使其美观。此版本要求 MT 管理器**大于等于 2.18.4 版本**。

2.目录

3.基本文件结构

```
#注释，由于word编辑器限制，全文高亮内容与实际不符，仅作参考
SignText=v1.3
@重要内容，带高亮，方便快速定位

!重要内容，不带高亮

!名称1:
    解释内容的分支1:
        "名称2"
        名称2由上被调出到等号解释
    解释内容的分支2:
        "名称1/1./2."
        上面是调用内容，相当于是名称1的分支1下的第二个(行)内容
        /1./2.就是路径，前面是主级的名称，不能用分支作为名称

!名称2=
    分支1=
        名称2由上面调出，还可以调用上面的内容
        "名称1/2./1.2.3."
        /2./1.2.3.是路径，代表名称1的分支2下的1,2,3行的内容
    分支2=
        "名称1/1./1."#无限调用
        相当于名称1的分支1下的第一行内容
        但此时内容就是当前分支2所在的名称2，所以当分支2被阅读时
        调用了名称2，而名称2中的分支2又调用了名称2，无限调用下去
    分支3='/storage/emulated/0/MT2/SignText - 标志文本v1.3.mtsx'
    分支4=
        上面的分支3引用了一个真实路径，表示需要从那个文件或文件夹读取信息
        路径也可以缩短，只要你知道文件位置或这个.st文件和那个文件在同一目录
        下
    分支5=
        1.1e+3 + 10^100 = ?
        匹配十进制数字，含浮点和科学计数法，数字紧挨文本表示为文本类型
        高亮的数字为数字类型，表示可以计算，上为算式写法

!名称3=
    v1=版本在解释内容中可以使用
    分支2=v1  也可在其后使用
    signtext=是标明语法版本的关键字，只在解释内容中使用
    其他关键字=all() any() none true false not and or
    十六进制颜色=*000000FF

类型:文本，数字，关键字，颜色，路径，版本
```

4.语法错误

由于 MT 管理器的文本编辑器并不是 IDE，并不会提示你存在语法错误，所以关于以下的语法错误，仅是作者规定的语法错误。在实际编写中，如果你出现了作者规定的语法错误，编辑器不会提示你，因为这只是一个普通文本编辑器，ST 也只不过是一个普通文本的带高亮版本而已，所以你可以放心大胆的随便写，只是可能会缺少一些美观感。

5.注释

注释由#开头，其后的内容不常规高亮，前面的内容无影响。仅有行注释。

注意：注释在" "和' '中不生效，仅作为对应的内容。

6.版本信息

在 SignText 语法中，英文关键字**不区分大小写**，所用到的字符也都是英文字符。第一行应注明此文件所使用 st 版本，例如：

```
SignText=v1.3
```

signtext 的类型是关键字，只在高亮的=或:前匹配，在普通文本中不匹配。它表示“标志文本”，用作标识这个语法的版本。它的前后不能有任何文字和_否则不匹配，如果仅用它作普通文本，可以在其前或后加上_符号。它应像上面一样使用，其他使用方法不推荐(视为**语法错误**)。

=的作用见 **8.解释内容**

v1.3 的作用见 **12.版本**。因为目前是 1.3 版本，所以填写 v1.3。

7.重要内容标记

在编写 ST 文本时，我们经常需要上下翻找，如果没有标记，我们就会很难找到我们需要的内容，所以可以使用@和!作为标记方便查找。

● @标记

@重要内容标记，只能在每一行的第一个位置，在@的前面可以有空格和制表符，后面的内容是普通文本。@标记具有行高亮功能，可以更快速找到文本，与 **8.解释内容**配合使用更好。

```
@@#第一个@高亮并带行高亮(编辑器限制未展现)且前面可以有空格，第二个@为普通文本
A   @#@不高亮，都是普通文本
```

● !标记

!重要内容标记，与@标记相比少了行高亮功能，其他一致。

8.解释内容

在 ST 中，解释内容是文件的主要部分，可以用:或=对需要的内容进行解释，下面的分支就像 if…then，if…elif…else 一样，用缩进决定层级，同一缩进是同一级。可以用@或!对它们进行标记。解释内容的类型是文本。

(1)冒号解释

if…then 样式:

```
早餐吃:
    包子:没吃完记得打包带走#包子相当于 if，冒号后面的相当于 then
```

为什么不能写在同一行？因为这样后面冒号部分不会高亮，成为普通文本。如果把冒号换成等号在同一行更不行。前面的就算可以，这也都是**语法错误**。

if…elif…else 样式:

```
倒计时:
    3 分钟:泡面#3 分钟是 if
    15 分钟:敷面膜#15 分钟是 elif
    30 分钟:看一下书#30 分钟是 else
```

在一个冒号解释的部分，里面不能出现等号解释，反之亦然。如把“3 分钟”后面的冒号换成等号，这样是**语法错误**。

高亮部分的前面可以有重要内容标记。高亮是从行首到:的，文本中间不能有空格制表符，否则成为普通文本。

```
aabb: #前面可以有空格
@   aabb:#前面@标记的后面可以有空格
a   abb:#文本中间有空格，普通文本
@   a  abb:#@标记可以，文本间有空格
```

这里需要注意，上面的 ST 写法是演示作用。在实际中，:=的前后或者下一层级应有内容，否则**语法错误**。只有一层解释的话可在:后面，也可在下一层级。

=与:的顺序是谁在前面就高亮谁。

```
ab=:
ab:=
```

可以看到，等号后面的:没有高亮，冒号后面也同样。

某些时候，我们希望匹配后面的=或:，这个时候我们就要用到转义符\。当解释部分的前面有@或!时，我们希望它不高亮，只是一个普通文本，这是我们就用\添加在它们前面，这样就不会高亮了。当我们希望解释部分里有冒号、等号、井号(注释)时，都可以在前面加上\，当然，如果在它们前面我们也只想用普通的\，就也加一个\。希望在其中有空格或制表符时，需要用\s(空格)或\t(制表符)表示。展示：

```
@abcd:   →\@abcd:     a bcd:   →a\sbcd:     a   bcd:   →a\tbcd:
abcd:=   →abcd\:=     abcd=:   →abcd\=:     abcd\:   →abcd\\:
abcd#:   →abcd\#:
```

signtext 在:和=中是关键字类型，具体见 **6.版本信息**。

v1.3 是版本类型，在:=中也会高亮，具体见 **12.版本**。

高亮部分后面解释的方法有两种，一种就是像上面一样的用普通文本解释，另一种是用" "调用到另一部分解释。具体见 **9.调用(调出)内容**。

(2)等号解释

等号解释与冒号是基本一致的，只有颜色不同，用于区分不同的部分。

9.调用(调出)内容

其实第一个没有缩进的:=前面你填写的内容就相当于变量名，它是可以被调用的，用" "进行调用。在同一行中，" "的前后可以有同样的调出内容" "和普通文本。

```
a.name:
    "b.name"
    任意内容 2
b.name=任意内容 1
```

对于 a.name 来说，它调用了 b.name 的内容到它的下一行，就相当于直接把“任意内容 1”放在了"b.name"的位置。对于"b.name"来说，由于它比最后一行的 b.name 来说出现的行数要早，所以就是"b.name"调出了 b.name，如果 a.name 是一个主要框架，不希望它有太多的内容，那么这个调出就使得原本可能要大量描述的地方就减少了内容。如果理解困难，它们其实就只是一个相互替换的效果吧。

强调一下，只有第一个没有缩进的:=前面的内容才相当于变量名，所以对这个位置的命名尽量简单不重复，重复了会导致调用错误(**语法错误**)。

那如果只想要单独调用一个部分里面的某些内容呢？拿上面的 ST 举例，如果你想要调用 a.name 里面的“任意内容 2”，可以写成`"a.name/2."`后面的/2.是路径功能，只能是/加除了 0 开头的数字加点(.)的格式，更长的可以写成/1./1./1./2. 等等。

如果想要多个内容，可以写成/1.2.3.4.等等，在这里如果有重复的视为找不到内容(**语法错误**)(例：/1.1.2.2.，这个暂时没法做到标记错误，单纯的找不到内容也是语法错误)，因为前面的已经调用。注意，在这里是绝对不能用多个分支下的内容的路径的(/1.2./1.2.)，可以的是/1./2./1.2.，这是一个单分支，因为多个分支下用这种方式已经无法表示出来。没有提到的其他错误写法都会被视为错误(**语法错误**)，如只有路径没有变量名。

这里有一个问题，如果下一层级还有:解释的内容，为什么不用下一层级:前的内容来作变量名呢？可以这么想，假如你写的内容特别特别多，如果子层级都是变量名，那么很快可用的变量名就会大量减少，导致后续非常麻烦。

还有一个特别重要的**逻辑问题**，不要出现我调用我自己的情况，还有你调用我，我调用你，这些都会导致死循环。

```
aa:"aa"#自己调用自己
a.a:"b.a"#相互调用
b.a="a.a"
```

" "和' '在:=中只表示文本类型，失去原本功能。" "可以在普通文本中对关键字进行转义。例如：`false` → `"false"` 这样 false 就表示为普通文本了。

在" "中的转义有这些：

```
"\\"  "\""  "\/"  "\@"  "\!"
```

其他\转义都是错误的(**语法错误**)，分别解释一下，\\是对\的转义，\"是在""中如果想有"就用这个，\/是对路径功能的转义，后两个是当@!在""的第一个位置时("@aabb")会标错，因为@!在开头时通过""调出时会变成由@!开头的重点标记，而重点标记不能由""进行标记，需手动标记。所以在这里注意：调用内容与解释内容有一定的呼应，所以它们的限制有部分相同。

10.路径

在文本中，我们可能需要查看此文件以外的文件或文件夹，或者这里的解释内容和路径中的文件有关，这时就需要通过路径来进行导向。路径一般在=:后面，且多个路径不应叠写在一行(**语法错误**)，应该写在相同的缩进的不同层级。路径是写在' '中的，由/开头，里面的内容任意，但是不能有\，\在路径中是非法的。

```
存档='/mc1.16.5/world1''/mc1.18.2/world1'#错误地叠写，应该如下
存档=
    '/mc1.16.5/world1'
    '/mc1.18.2/world1'
    '/minecr\aft'#包含\
```

任意不由/开头或者里面含有\的路径都**错误**。它只有\t 和\'的转义。路径' '的前后最好不要有普通文本和" "。

```
照片=
    '1.jpg'
    '/\t\''
```

这个路径并不是严格规范具体路径，路径的长短可以根据它是否与此 st 文件在同一文件夹下或者你对那个文件的位置很熟悉就可以简写。如果是在部分相同的文件夹下就可以把相同的文件夹部分给省略。

11.数字

没有高亮的数字表示为普通文本类型，高亮的数字是数字类型，表示可以运算。ST 支持正负数、浮点数、科学计数法、指数，仅支持**十进制数**。

```
a=-1,+1,-1.1,-1.1e1,-1.1e-1,-1^-10
b=36 * (64 - 42)^2 - (-16)#算式
```

12.版本

版本原本是为 signtext 专用的版本标记格式，但也可作其他版本标识。在:=和普通文本中通用的版本标识格式为由 v 加数字，最高 4 个数字，中间加点，其后只能跟空白符、冒号、等号、下划线、逗号，如果出现多余的点则不会标识。在:=中，版本标识不能作为变量名，否则**语法错误**。在普通文本中，其前不能是*，因为它是**颜色**的标识符。

```
a=v1,v1.1.1.1,v1_
b=
    v1.#多余的点
    v2=*v1#由*开头
```

13.关键字

在 ST 中，关键字有 all,any,none,true,false,not,and,or。这些关键字都不在解释内容中匹配，只在普通文本中匹配。signtext 是特殊的关键字，在之前已经介绍了，这里就不再叙述。在普通文本中，可以用" "将关键字标记为普通文本。最好将这些关键字单独使用。

`all()` 后面必须接括号，否则**语法错误**，全部正确则输出正确，判断括号里的内容。

`any()` 同上，任意一个正确则输出正确。

`none` 表示无或空

`true` 表示正确

`false` 表示错误

逻辑关键字：not,and,or。优先级如此，是自然语言的判断，可用()调整优先级。

`not` 非　`and` 与　`or` 或

14.颜色

由*开头，后面接一串十六进制数，是 HEXA 的格式，即 RGB,RGBA,RRGGBB,RRGGBBAA。例如：*FFFFFF 表示白色，呈现在背景。不会在解释内容中匹配，前后可以有空格，逗号。

15.类型

ST 的类型有：文本类型、数字类型、关键字、颜色类型、路径、版本类型。

16.安装语法高亮文件

将下载的压缩包解压后，在 MT 管理器里点击 SignText – 标志文本 v1.3.mtsx，点击安装即可安装完成。接下来创建.st 文件你就能愉快的写内容和高效地阅读啦！