用Lua扩展谷歌拼音输入法

sherrywasp 2015-08-16 原文

谷歌拼音输入法最后一次更新是2013年，最近2年毫无动静，这个产品应该已经停了，不过这并不影响对它的使用，我一直喜欢它的简洁和稳定。

说不上来什么原因，忽然想起了摆弄摆弄谷歌拼音输入法的扩展特性（我经常会有这种不明原因地忽然想折腾点什么的状况）。于是搜索到官方文档——“谷歌拼音输入法-扩展API开发指南”，粗略阅读了一番。考虑到因为某些不可抗拒的力量，这份文档不会很稳定的存在，所以自己整理一下，权当笔记。

扩展其实很简单，利用API，往谷歌拼音里注册自己写的函数，然后就可以在打字的时候调用了。扩展是基于 **Lua** 这门脚本语言的（话说 Lua 确实是全球各种扩展的常客），扩展包就是一份 Lua 脚本文件，然后可以通过输入法的属性窗口进行添加扩展操作。总共分为三种途径：命令扩展、整合扩展、转换器扩展。不管是哪一种，其输出都是以候选项的形式呈现。

所谓命令扩展：在谷歌拼音中，有一种 **i** 扩展模式，即首字母输入 i 后，可以得到一些内置的特别功能输出，比如输入 **rq** 可以输出当前日期，输入 **sj** 可以输出当前时间，还可以掷个骰子，画个 Google 主页等等（注：这种字符画必须在“等宽字体”下才看得出效果）。这种特性就叫命令扩展，输入字母 i 相当于切换到了谷歌拼音的命令模式，我们可以添加自己的命令，实现自己的需求。有一点需要注意的是——命令扩展的触发关键字，只支持2个字符，意思是说，你只能规定一个包含2个字符的字符串作为触发事件，当你进入谷歌拼音 i 模式时，输入你所设定的2个字符，则可自动调用你写的函数。

所谓整合扩展：就是以用户输入的字符或者是谷歌拼音显示出来的候选字词来进行匹配，触发被注册的函数。比如说我们写了一个函数，返回字符串“hello, world”，将这个函数注册成一个整合扩展，并设置触发的关键字为“hello”。那么，当我们用谷歌拼音输入“hello”这5个字母时，输入法的候选项首页最后一项将是我们的函数返回值“hello, world”。这种方式和命令扩展的区别是：不需要输入 i 切换模式，直接按照正常的输入方式打字即可调用自定义函数。

所谓转换器扩展：就是针对候选项做变换，比如把出现的候选项字词，每一项前后加上“*”号

用来装饰一种效果等。其工作原理就是把当前页的每一项候选项作为参数，传递给我们写的函数，然后经过处理再返回并替代原候选。

这三种扩展途径的具体使用说明附在文末。

接下来需要说一下返回值。

之前说了，不管哪种方法，最终都是以候选项的形式返回输出。但这个候选项返回可以不止一个，比如函数返回的是一个列表，那么出现在输入法界面上的就会是多个候选项，等于我们自己给自己制造候选。还有一种情况，我们通常得到的候选项都是一个字、一个词、一句话，但其实我们还可以得到一段多行文本，只需要在自定义函数中返回一段带换行符的字符串即可。

了解了谷歌拼音输入法的扩展特性之后，我随即写了一个扩展包，里头包含了一些我临时想到的功能。

由多行文本输出（在谷歌拼音里叫<字符画>），我想到了这个可以用来做自己的代码片断（snippet）。把自己喜欢的，常用的代码片断做到自己使用的输入法里，这样，不管在哪，不管切换到了什么文本编辑器或者IDE里，只要输入法是谷歌拼音，用上自己的扩展包，你就有了统一的代码片断功能了。

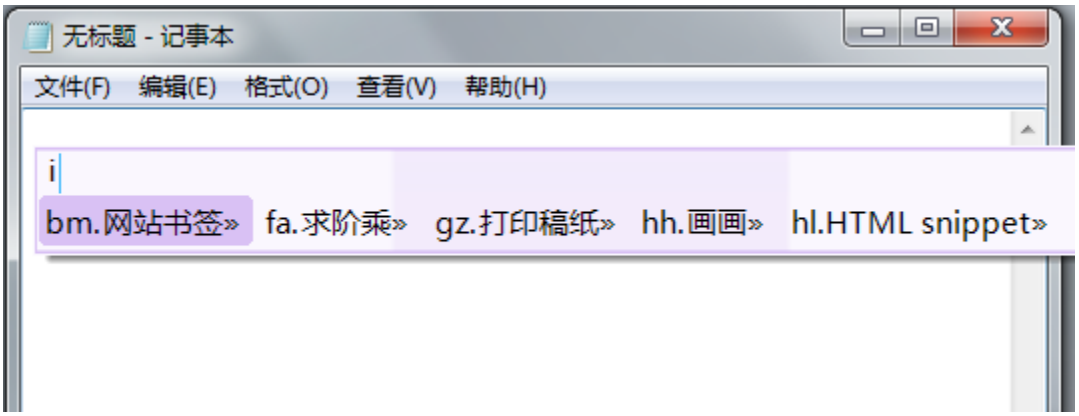
比如，我写了一个函数，返回一段HTML标记，如下所示：

```
--插入一段HTML基本标记
function html_snippet()

    output = [[
<!DOCTYPE html>
<html>
<head>
    <title>Untitled</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
    <link rel="stylesheet" type="text/css" href="">
    <script type="text/javascript" src=""></script>
</head>
<body>

</body>
</html>]]

    return output

end
```

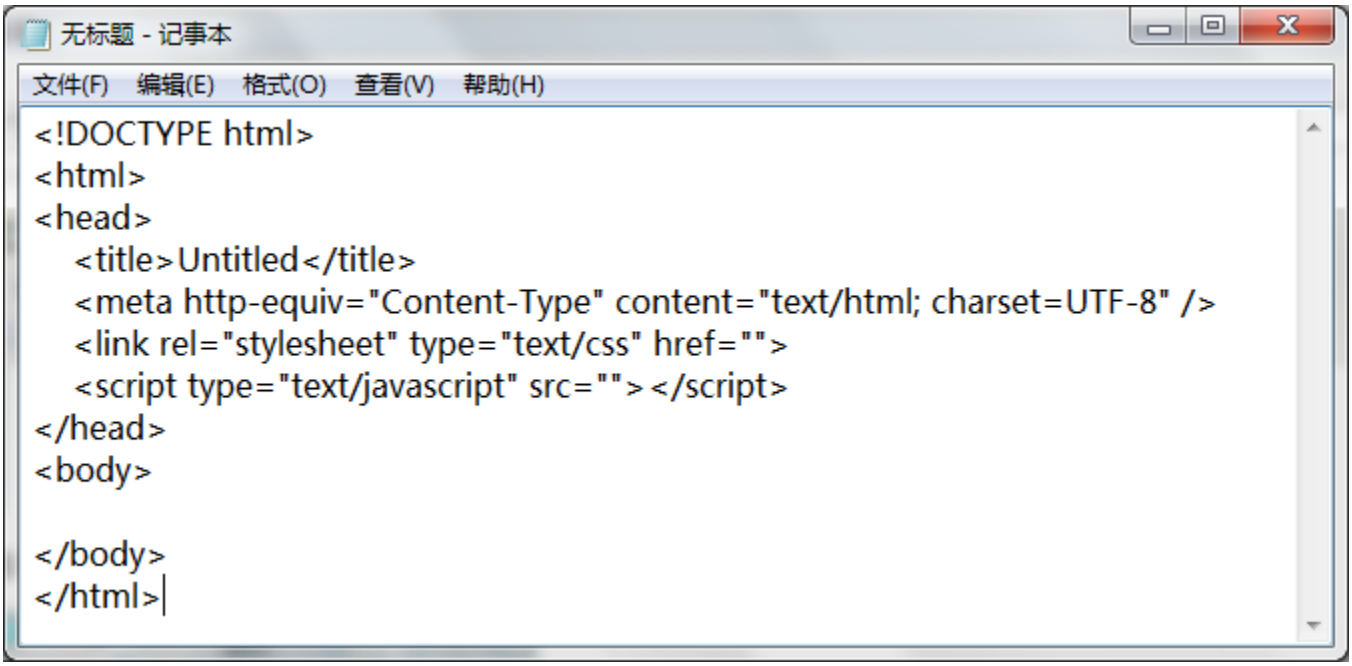
然后将这个函数注册到命令模式中

```
1.  ime.register_command('hl', 'html_snippet', 'HTML snippet')
```

这样，当在谷歌拼音输入法中，输入 i 进入命令模式中之后，可以在候选项里看到一个名为“HTML snippet”*的候选项，如图所示：*



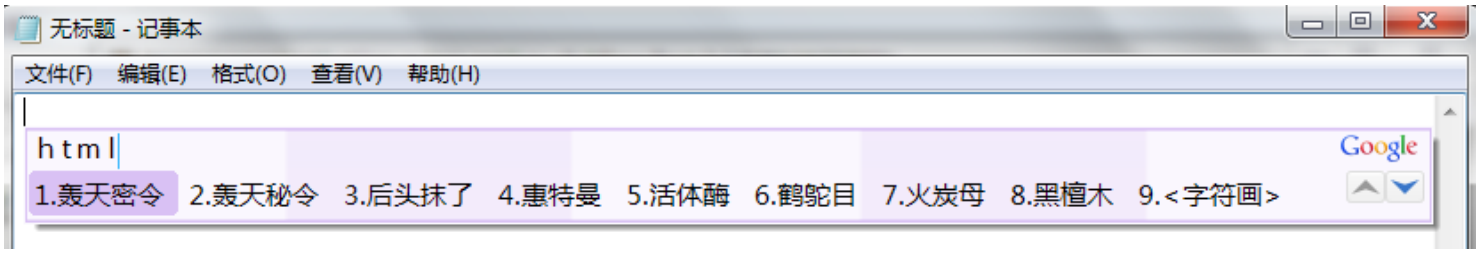
键入预设的关联字符“hl”，*然后点击空格，屏幕上会立刻输出一段连格式都排好了的*HTML*标记。*



同时，我将这个函数继续用整合扩展的方式进行注册，代码如下：

```
ime.register_trigger('html_snippet', 'HTML snippet', {'html'}, '')
```

之后，除了通过 i 命令模式实现快速插入这段HTML标记的效果外，直接在输入法中键入“html”字符，即可在首页最后一个选项得到所需要的输出。



键入对应的序号9，记事本中也会打出一段和上图一样的HTML标记文本。

有了这样的扩展，你就算是在记事本里也可以玩插入代码片断功能了。

因为谷歌拼音是基于Lua这门语言来扩展，自然也包含了一些可以调用的Lua内置函数、模块等，比如数学函数。

以下这段代码就是往谷歌拼音输入法里添加数学计算功能的代码：

```
-- 求平方
function square(x)

    return y * y
end

-- 求平方根
function sqrt(x)

    return math.sqrt(y)
end

-- 注册 求平方
ime.register_command('sq', 'square', '求平方', 'none', '输入一个数，得到其平方')
-- 注册 求平方根
ime.register_command('sr', 'sqrt', '求平方根', 'none', '输入一个正数，得到其平方根')
```

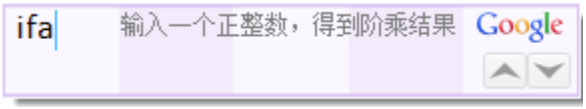
需要注意的是，这种带数字参数的函数，其参数是通过键入数字字符传递的，又因为输入法

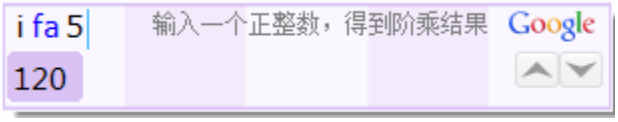
默认会把数字字符作为选择候选项的一种操作，所以，在注册这种扩展时，需要告诉输入法关闭数字选择功能，也就是上面注册代码中，函数的第4个参数设置为'none'。

求完平方和根，顺手写了一个求阶乘的扩展：

```
-- 求阶乘
function factorial(argument)

      then
          )
     else
         return x
     end

end

-- 注册 求阶乘
ime.register_command('fa', 'factorial', '求阶乘', 'none', '输入一个正整数，得到阶乘结果')
```

阶乘计算示例：





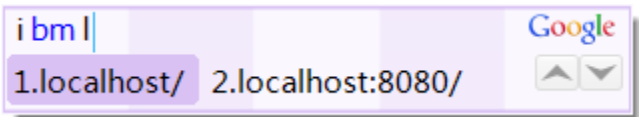
还可以把输入法扩展为网站书签（虽然这么做的意义.....）

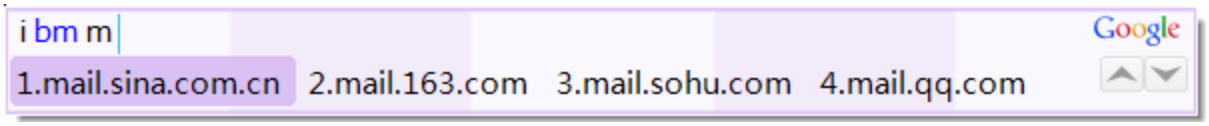
代码如下：

```
-- 网站书签
_BOOKMARK_TABLE = [[
b cn.bing.com
```

```
ba www.baidu.com
c www.cnblogs.com/
g www.google.com
l localhost/,localhost:8080/
m mail.sina.com.cn,mail.163.com,mail.sohu.com,mail.qq.com
p www.python.org
t www.taobao.com,www.tmall.co
tw twitter.com
w wikipedia.org
y search.yahoo.com
]]

_BOOKMARK = ime.parse_mapping(_BOOKMARK_TABLE, "\n", " ", ",")

function bookmark(input)
      return _BOOKMARK[input]
end

-- 注册 网站书签
ime.register_command('bm', 'bookmark', '网站书签')
```

这样，当输入“i”进入命令模式，然后键入“bm”，调用网站书签函数之后，我们就可以通过键入自定义的快捷键，得到网站地址，如图所示：





以上就是利用 Lua 对谷歌拼音输入法进行扩展的一个简介。其实谷歌拼音相当于集成了一个 Lua 的子集运行环境，对它的扩展全凭在该子集下对 Lua 的把玩了。

附：

谷歌拼音输入法注册扩展方法的详细语法介绍（以下内容直接摘自官网文档）

命令扩展的语法：

ime.register_command(command_name, lua_function_name, description, leading, help)

ime*是提供给*Lua*脚本使用的，与输入法内核交互的专用模块。*register_command*是向谷歌拼音输入法注册新的*i*扩展模式命令扩展所使用的函数。函数的各参数含义如下：*

command_name

*2字符长的字符串，必须由两个英文字母（*a-z*）组成。定义了要注册的命令名字。如果新注册的命令名称和此前已经注册的某个命令重名（判断重名时不区分大小写），则*register_command*函数调用失败，新命令扩展无法注册到输入法中。*

lua_function_name

*字符串。给出此命令在*i*扩展模式中运行时对应的*Lua*入口函数。这必须是一个已经存在的，接收一个或零个参数的*Lua*函数。*

description

*字符串。命令的简短描述。此描述会显示在*i*扩展模式的命令选择界面中，向用户简要说明某命令的功能。不要使用太长的简短描述，一般不要超过*10*个字符。*

leading [*可省略***]**

字符串。用户选择此命令的候选项目时，可以使用的快捷键，可以是以下三个特定字符串之一：

"**digit**": *默认值。表示用*1, 2, 3, ...*这样的数字作为候选项选择键。*

"**alpha**": *表示用*a, b, c, ...*这样的英文字母序列作为候选项选择键。*

"**none**": *表示不使用候选项选择键。*

*注：默认情况下，输入法使用*1, 2, 3, ...*数字键作为候选项选择键。但是，当*i*扩展模式的某个命令希望接收数字*1, 2, 3, ...*作为自己的参数时，为避免冲突，就不能使用*"digit"*方式的候选项选择键了。同理，当命令希望接收包含英文字母的参数时，就不能使用*"alpha"*作为候选项选择键。*

help [*可省略***]**

*字符串。比*description*略长的帮助信息，但一般不要超过*50*个字。当用户键入了*"i"*以及特定的命令名后，输入法候选窗口的右上方会显示此文字信息，用于提示用户如何输入后续参数。*

整合扩展的语法：

ime.register_trigger(lua_function_name, description, input_trigger_strings, candidate_trigger_strings)

ime*是提供给*Lua*脚本使用的，与输入法内核交互的专用模块。*register_trigger*是向谷歌拼音输入法注册新的整合扩展所使用的函数。函数的各参数含义如下：*

lua_function_name

*字符串。给出此扩展运行时对应的*Lua*入口函数。这必须是一个已经存在的，接收一个参数的*

Lua函数。

description

字符串。扩展功能的简短描述，向用户简要说明某扩展的功能。不要使用太长的简短描述，一般不要超过10个字符。

input_trigger_strings

一个字符串组成的Lua列表，包含零个或多个特定的由英文字母或通配符*组成的字符串。这里给出的所有字符串在输入法运行时将分别与用户的输入内容匹配，一旦用户的输入和给出的某个特定字符串相同（或使用通配符匹配成功），注册的扩展函数就会被调用，扩展函数返回的候选项结果将会被插入到输入法的候选项列表中。

candidate_trigger_strings

一个字符串组成的Lua列表，包含零个或多个特定的由英文、中文、数字等可显示字符或通配符*组成的字符串。这里给出的所有字符串在输入法运行时将分别与输入法得到的候选项进行匹配，一旦候选项列表第一页中某个候选项和给出的某个特定字符串相同（或使用通配符匹配成功），注册的扩展函数就会被调用，扩展函数返回的候选项结果将会被插入到输入法的候选项列表中。

关于通配符匹配：input_trigger_strings和candidate_trigger_strings中的字符串可以在开头或结尾包含通配符*，表示前缀匹配或后缀匹配。例如：

abc*

表示匹配前缀为abc的任意字符串。例如，字符串abc，abcd，abcde都可以与之成功匹配。

*abc

表示匹配后缀为abc的任意字符串。例如，字符串abc，dabc，deabc都可以与之成功匹配。

使用ime.register_trigger注册整合扩展时，请注意以下几点：

参数input_trigger_strings和candidate_trigger_strings不能同时为空表。

输入法在激活整合扩展函数时，将优先匹配input_trigger_strings，然后再匹配candidate_trigger_strings。匹配candidate_trigger_strings时，会按照输入法得到的候选项顺序依次尝试。对每一次输入，一旦找到了匹配，就只插入该匹配对应的扩展函数返回的候选项结果，不再继续尝试其他匹配。

虽然扩展函数可以返回一个或多个结果，但对于整合扩展来说，目前只有第一个候选项结果会被插入到输入法的候选项列表中。

目前一个整合扩展可以通过input_trigger_strings和candidate_trigger_strings注册的字符串数目是有限制的，一般不要超过200个。

目前整合扩展在匹配candidate_trigger_strings时，只会与候选项列表第一页中的候选项进行匹配。

整合扩展的入口函数一般应接收一个参数。在激活整合扩展函数时，输入法会把激活整合扩

展函数的字符串（或者是用户输入的内容，或者是某个特定的候选项）作为唯一的参数传递给入口函数。这样，注册了多个匹配字符串的整合扩展函数就可以在被调用时通过参数知道究竟是哪个字符查激活了自己。当然，在不需要时，入口函数也可以简单地忽略这个参数。

转换器扩展的语法：

ime.register_converter(lua_function_name, description)

ime是提供给Lua脚本使用的，与输入法内核交互的专用模块。register_converter是向谷歌拼音输入法注册新的转换器扩展所使用的函数。函数的各参数含义如下：

lua_function_name

字符串。给出此扩展运行时对应的Lua入口函数。这必须是一个已经存在的，接收一个参数的Lua函数。

description

字符串。扩展功能的简短描述，向用户简要说明某扩展的功能。不要使用太长的简短描述，一般不要超过10个字符。对于转换器扩展，此描述信息会被输入法的用户界面显示给用户，以便选择特定的转换器扩展。

用户开启转换器时，输入法会依次将每个候选项作为参数调用转换器对应的Lua入口函数。也就是说，对于每个候选项，Lua入口函数都被调用一次。

转换器扩展对应的Lua入口函数应当返回且只返回一个结果，即返回对原候选项进行变换后的新候选项。如果不希望变换某个候选项，可以将输入参数的值直接返回。如果没有返回任何结果，或返回的结果数目多于一个，则输入法认为该扩展函数没有对候选项做任何变换。

安装了转换器扩展后，用户可以从输入法的用户界面启动或关闭特定的转换器。

官方文档链接

http://www.google.com/intl/zh-CN/ime/pinyin/api.html