Google拼音输入法 API 开发指南

摘要：为了帮助开发者在谷歌拼音输入法的基本输入功能基础上，开发和定义更丰富的扩展输入功能，谷歌拼音输入法提供了以Lua脚本编程语言为基础的输入法扩展API。利用输入法扩展API，开发者可以编写自定义的输入功能，并将脚本分享给谷歌拼音输入法的用户安装、使用。

新增功能

- 输入法扩展API支持转换器扩展，在用户开启转换器时，可以对候选项做诸如装饰、特效、变换等操作。
- 新增一组用于UNICODE编码转换的字符串实用函数。

入门

为了帮助开发者在谷歌拼音输入法的基本输入功能基础上，开发和定义更丰富的扩展输入功能，谷歌拼音输入法提供了以Lua脚本编程语言为基础的输入法扩展API。利用输入法扩展API，开发者可以编写自定义的输入功能，并将脚本分享给谷歌拼音输入法的用户安装、使用。

一段简单的Lua脚本程序就可以构成一个最基本的输入法扩展模块。下面是Hello,World!程序示例：

helloworld.lua

```
function HelloWorld()
  return "Hello,World!"
endime.register_command("hw", "HelloWorld", "test")
```

这一段代码由一个自定义的Lua函数和一行ime.register_command函数调用组成。自定义的Lua函数HelloWorld()简单地返回一个Lua字符串"Hello,World!"，这表明该输入法扩展函数被调用后，显示给最终用户的候选项为"Hello,World!"。ime.register_command函数调用将自定义函数注册为谷歌拼音输入法的一个命令扩展。其中，第一个参数"hw"表示该命令扩展在i扩展模式中对应的命令是"hw"，第二个参数表示该命令扩展对应的入口函数（自定义的Lua函数）是"HelloWorld"，第三个参数是显示在i扩展模式命令列表内的简短说明文字。

使用任何文本编辑器输入上述程序后，以helloworld.lua为文件名保存到安装有谷歌拼音输入法的计算机中。然后，打开谷歌拼音输入法选项设置窗口，在"扩展"页面中，点击"安装扩展包"按钮，选择保存在计算机内的helloworld.lua（也可以从Windows资源管理器，右键单击helloworld.lua文件，选择安装到谷歌拼音输入法）。安装后，打开记事本程序，切换到谷歌拼音输入法，键入"ihw"，谷歌拼音输入法的候选项窗口中将出现唯一候选项"Hello,World!"。

除了显式用"ihw"这样的命令来激活扩展函数以外，扩展函数还可以由用户在使用拼音输入法时输入的特定内容或特定候选词激活。例如，在上述helloworld.lua最后添加一行：

```
ime.register_trigger("HelloWorld", "test", { "hello" }, {})
```

这一行的作用是将函数"HelloWorld"注册为谷歌拼音输入法的一个整合扩展。第一个参数是扩展对应的入口函数"HelloWorld"，第二个参数是简短说明文字，第三个参数给出希望将扩展关联到哪个或哪几个用户输入串（这里是字符串"hello"），第四个参数给出希望将扩展关联到哪个或哪几个特定的候选词（这里是空表，表示不关联）。

打开谷歌拼音输入法选项设置窗口，在"扩展"页面中，使用"移除扩展包"按钮将刚才安装的helloworld.lua删除，然后重新安装更新后的helloworld.lua。打开记事本程序，切换到谷歌拼音输入法，键

入"hello"，谷歌拼音输入法的候选项窗口中，除了出现通常的中文英文候选词提示外，还将出现由整合扩展函数返回的候选项"Hello,World!"。

事实上，谷歌拼音输入法提供的输入法扩展API可以用来开发各种不同的输入体验，例如，根据用户输入的参数返回相应的信息内容，查表输入特定的文字信息，完成自定义的甚至包含随机变量的计算并以不同形式返回结果，将用户刚刚输入的文字内容转换为另一种表现形式，等等。谷歌拼音输入法提供的i模式的缺省功能，包括时间和日期格式转换，查星座，掷骰子，打印字符等，都是一些最简单的示例。

本指南的后续内容详细介绍了开发输入法扩展所需要的各种知识。我们也鼓励开发者直接参考已有的示例程序。例如，i扩展模式的缺省功能是由安装在以下位置的Lua脚本程序实现的：

```
XP: C:\Documents and Settings\All Users\Application Data\Google\Google Pinyin 2\Extensi
ons\base.luaVista /
Windows 7: C:\ProgramData\Google\Google Pinyin 2\Extensions\base.lua
```

三种不同的扩展方式

谷歌拼音输入法扩展API提供了三种扩展拼音输入法的方式：**命令扩展**、**整合扩展**与**转换器扩展**。

- **命令扩展**：将脚本程序中的某个入口函数关联到一个两字母长的自定义命令。当用户先键入i然后键入该命令时，输入法即激活该扩展函数，然后在候选项列表中显示扩展函数返回的候选项结果。
  - 命令扩展适用于那些用户明确希望在特定场景下使用特定输入功能，且候选项较多，或较复杂的情况。例如，用户明确希望根据生日查询并输入星座信息等。
- **整合扩展**：将脚本程序中的某个入口函数关联到特定的键盘输入串，或特定的中英文候选项。当用户使用拼音输入法时，一旦用户通过键盘输入的字符串与整合扩展关联的特定字符串（可包含通配符）匹配，或拼音输入法解析出的某个候选项与整合扩展关联的特定字符串（可包含通配符）匹配，输入法即激活该扩展函数，并将扩展函数返回的候选项结果插入到候选项列表中。
  - 整合扩展适用于那些在不妨碍用户正常输入的情况下，根据当前输入或候选内容，插入少数相关候选项的情况。例如，用户在输入中文时间的同时，也可能希望直接输入当前时间，这时，整合扩展直接把扩展函数返回的当前时间整合至候选项列表中，就显得比较方便了。
- **转换器扩展**：将脚本程序中的某个入口函数注册为一个特定的转换器。当用户通过输入法的用户界面（如功能菜单）开启该转换器时，输入法产生的每个候选项被依次当做参数送入该转换器函数，进行运算后，函数返回的结果将会替换掉原候选项的内容，被输入法显示在相应位置。
  - 转换器扩展必须由用户主动开启。一旦开启，就会应用于所有候选项。因此，转换器扩展适用于为所有候选项增加装饰、特效，或者对所有候选项按规则进行变换的情形。例如，为候选项或候选项的每个字增加星号修饰，直接在候选项的每个字后面输出该字对应的Unicode编码，将简体汉字变为繁体汉字，等等。

下表对不同的扩展方式进行简单的对比：

扩展方式	命令扩展	整合扩展	转换器扩展
注册方式	ime.register_command (...)	ime.register_trigger (...)	ime.register_converter (...)
适用范围	用户明确希望在特定场景下使用特定输入功能，且候选项较多，或较复杂的情况	在不妨碍用户正常输入的情况下，根据当前输入或候选内容，插入少数相关候选项的情况	为所有候选项增加装饰、特效，或者对所有候选项按规则进行变换的情形
应用实例	根据输入的生日查询星座；列举并输入特定的字符画	用户输入中文时间时，在候选项列表里插入当前时间；用户输入中文哈哈时，在候选项列表插入相应的表情符号	为候选项或候选项的每个字增加星号修饰；直接在候选项的每个字后面输出该字对应的Unicode编码；将简体汉字变为繁体汉字
激活方式	用户输入i加2字符长的命令，激活相应的命令扩展	用户输入的拼音字符串或输入法产生的某个候选项与整合扩展关	用户从输入法的用户界面（如功能菜单）开启特定的转换器，激活相

扩展方式	命令扩展	整合扩展	转换器扩展
		联的特定字符串（可包含通配符）匹配时，激活相应的整合扩展	应的转换器扩展

注册命令扩展

在Lua脚本中，向谷歌拼音输入法注册一个命令扩展的基本语法是：

```
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。

lua_function_name给出的命令入口函数可以接收一个或零个参数，例如：

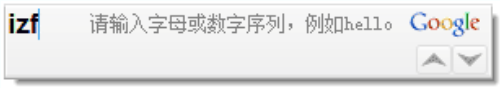
```
function my_entry_function()  -- 做某些处理并返回结果end
```

当入口函数接收一个参数时，输入法会把用户在i扩展模式中i+两字母命令名输完后继续输入的所有内容作为一个字符串参数，传给入口函数。例如，用户先后键入ihw123，则，用户激活的命令名是hw，输入法调用该命令对应入口函数时，以字符串方式传入参数123。入口函数可以对参数进行运算处理，并返回对应的结果。例如：

```
function my_entry_function(argument)  -- 将参数argument转换为数字，计算并返回其平方根...end
```

注册命令扩展时传入的提示信息description会在用户看到i扩展模式的命令列表时显示，以提示用户该命令的功能。这个字符串应当尽量简短（不超过10个字符）。如下图中的掷骰子，打印字符等，都是description：

注册命令扩展时传入的提示信息help会在用户选中了某特定命令后，显示在输入法候选窗口的右上角。这个字符串可以比description略长，但一般也不要超过50个字符。例如，下图中用户选择了打印字符命令zf后，显示出来的请输入字母或数字序列，例如hello就是help的内容：



注册整合扩展

在Lua脚本中，向谷歌拼音输入法注册一个整合扩展的基本语法是：

```
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。

注册转换器扩展

在Lua脚本中，向谷歌拼音输入法注册一个转换器扩展的基本语法是：

```
ime.register_converter(lua_function_name, description)
```

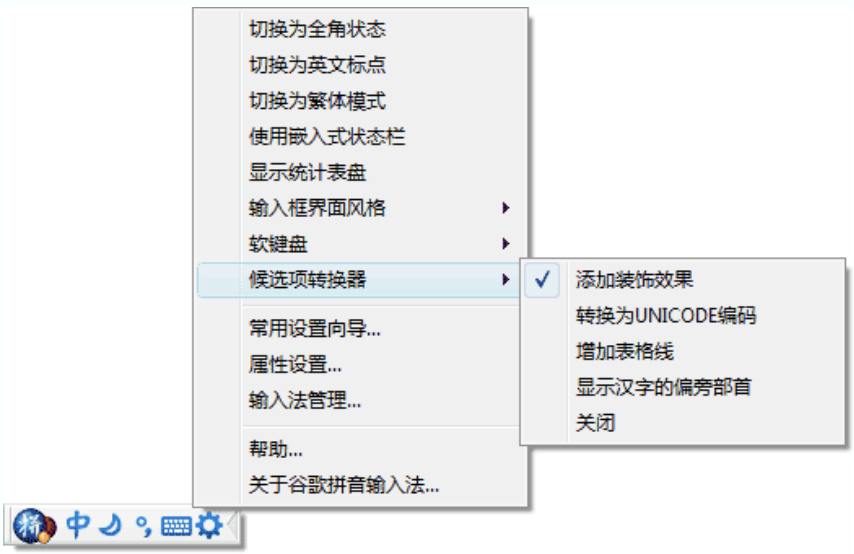
ime是提供给Lua脚本使用的，与输入法内核交互的专用模块。register_converter是向谷歌拼音输入法注册新的转换器扩展所使用的函数。函数的各参数含义如下：

- **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。如下图，从功能菜单开启或关闭特定的转换器：



返回候选项

一般的，扩展对应的入口函数可以返回一个或多个候选字符串。命令扩展会显示所有返回的候选项，整合扩展目前只会将第一个候选项结果插入到输入法的候选项列表中，而转换器扩展则只接受返回一个候选字符串的入口函数。

返回的参数类型可以是Lua字符串类型，也可以是Lua数字类型，还可以是Lua布尔类型。但这些返回值

返回输入法内核后，都会被转换成字符串显示给最终用户，以便用户选择输入。

要返回唯一的候选字符串，只要直接使用return语句返回字符串、数字或布尔值即可，例如：

```
function TestString(argument)
 -- 做某些处理  return "a string"endfunction TestNumber(argument)
 -- 做某些处理  return 1234.56789endfunction TestBoolean(argument)
 -- 做某些处理  return trueendfunction TestAnotherBoolean(argument)
 -- 做某些处理  return 3 > 2end
```

要返回两个或更多结果，只要返回一个Lua的列表对象即可，例如：

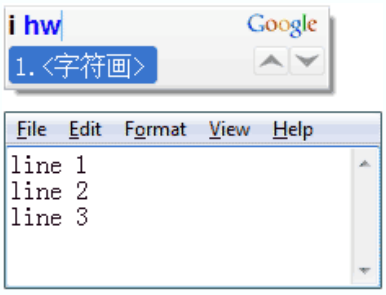
```
function TestTable(argument)  return {"abc", "def", "ghi", 123, true}end
```

列表中的每个元素可以是Lua字符串，数字或布尔值，但不能嵌套列表。列表中的每个元素将对应于输入法显示给用户的候选项列表中的一个候选项。上面这个函数返回的列表在输入法中的显示如下图所示：

除了单行字符串结果外，命令扩展的入口函数还可以返回一个或多个包含换行符的多行结果（其他扩展方式，如整合扩展和转换器扩展，目前不建议返回多行的结果）。换行符在Lua程序的字符串常量中用"\n"表示。例如下面的函数：

```
function TestMultilines(argument)  return "line 1" .. "\n" .. "line 2" .. "\n" .. "line
3"end
```

多行结果在输入法的候选窗口中被显示为<字符画>，当用户选择输入该候选项后，多行文本被插入到用户当前文档中，如下图：



返回提示信息

对于命令扩展，当用户刚输入完扩展模式的命令名称，尚未输入命令参数时，入口函数将被调用，此时传给入口函数的参数为空字符串。这时，入口函数可以通过返回一个提示信息表，来提示用户有几种预定义的候选参数，并在输入法的候选窗口中，允许用户直接选择某个预定义参数。例如：

```
function TestMetatables(argument)
 if #argument == 0 then
 -- 如果没有参数，则返回提示信息表，以便用户直接选择预定义的参数"num"或"chs"
 return { { suggest = "num", help = "数字123" },
 { suggest = "chs", help = "中文一二三" },
 }
 elseif argument == "num" then
 -- 如果参数是"num"（可能是用户键入的，也可能是用户根据提示信息表直接选择的），则返回数字结果
 return 123  elseif argument == "chs" then
 -- 如果参数是"chs"（可能是用户键入的，也可能是用户根据提示信息表直接选择的），则返回中文结果
 return "一二三"  endend
```

返回的提示信息表必须符合上述格式，即，表中的每个元素都是一个子表，每个子表内有两个元素：键名suggset的元素表示要提示用户输入的一个候选参数，键名help的元素表示对该参数的简短说明文字（不要超过10个字符）。上述入口函数在用户没有输入参数时，输入法显示的提示窗口如下图：

这时，用户可以试用上下键，翻页键和鼠标选择自己要输入的参数，也可以直接用键盘输入。

其他扩展方式，如整合扩展和转换器扩展，不支持提示选择参数功能，它们将忽略入口函数返回的此类

提示信息。

使用ime模块

在开发者编写的Lua脚本中，代码除了可以调用Lua本身提供的各模块功能（是标准Lua运行环境所提供功能的一个子集），还可以使用ime模块访问输入法的相关信息，以实现与输入法有关的特定功能。

例如，可以使用ime模块的get_last_commit()函数获得用户上一次键入的字符串，并根据字符串的内容进行相应的计算，返回特定结果。比如，用户键入了你好，然后使用i扩展模式中的某个功能，该功能看到你好后，自动返回hello。实现这一简单逻辑的代码如下：

```
function TestConvertHello()
 if ime.get_last_commit() == "你好" then
 return "hello"  else    return "not found"  endend
```

更详细的信息请参见API参考一节中有关ime模块的部分。

错误处理

Lua入口函数接收无法处理的参数，或者发生其他内部错误时，可以简单地不返回任何参数，或者使用Lua语言内置的error()函数向输入法报告错误信息，例如：

```
function TestIgnoreError(argument)
 if #argument > 5 then
 return
 end  return 123endfunction TestReportError(argument)
 if #argument > 5 then
 error("argument length > 5")
 end  return 123end
```

使用error()函数报告的错误信息不会在输入法用户界面中显示，但可以使用控制台工具测试脚本程序并查看错误信息。参见下面的开发与调试一节。

开发和调试

输入法扩展脚本程序可以使用任何源代码/文本编辑器创建。在将输入法扩展包安装到谷歌拼音输入法之前，可以使用控制台工具测试扩展程序，以确保程序功能正确。请从以下链接下载用于开发调试谷歌拼音输入法扩展脚本的控制台工具：

下载控制台工具 (GooglePinyinApiConsole.exe)

在命令行运行控制台工具GooglePinyinApiConsole.exe时，需要给出的命令行参数是一个或多个待测试的脚本文件路径，控制台工具会加载所有指定的输入法扩展脚本，并启动一个交互式界面，供开发者测试执行扩展模式。例如：

```
GooglePinyinApiConsole.exe ext1.lua ext2.lua ext3.lua
```

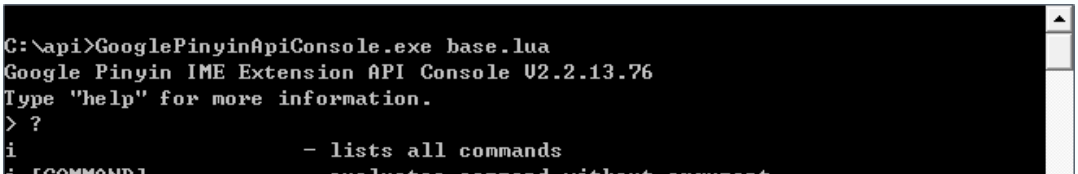
在控制台工具的交互式界面中，键入"help"，可以查看帮助信息：

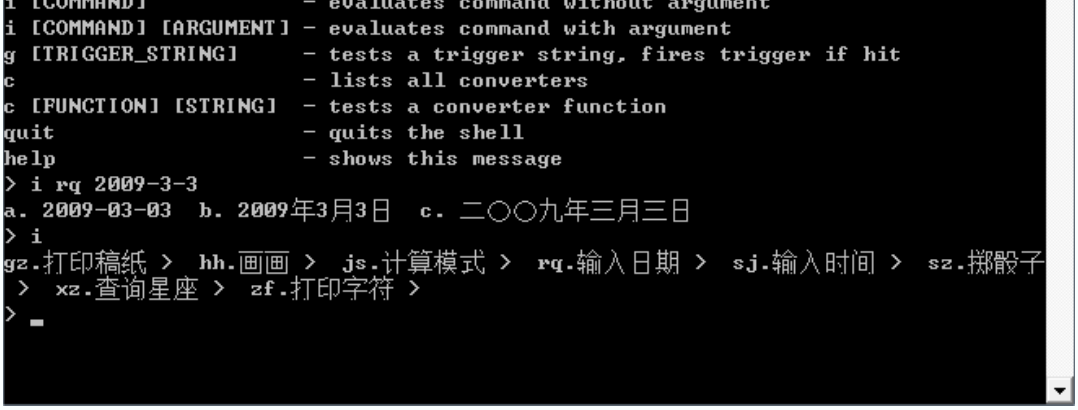
i - 列出所有已注册的命令扩展i [COMMAND] - 无参数执行某命令扩展i [COMMAND] [ARGUMENT] - 有参数执行某命令扩展g [TRIGGER_STRING] - 尝试利用某字符串参数激活整合扩展c - 列出所有已注册的转换器扩展c [FUNCTION] [STRING] - 测试转换器函数quit - 退出控制台工具help - 显示帮助信息

当脚本加载或执行过程中发生错误时，控制台工具会打印显示错误信息。错误信息包括发生错误的脚本文件名，错误行号，错误内容等。

控制台工具并不是真实的输入法运行环境，因此，一些和输入法特定功能相关的接口，例如，ime.get_last_commit()，在控制台运行的脚本中无法得到实时的输入法相关信息，这时，此类函数的返回值是简单的固定字符串，如测试。整合扩展也无法像在输入法中那样，将结果插入在输入法的候选项列表中。

下图显示了使用控制台工具测试缺省输入法扩展包中各扩展功能的情形：





关于Lua语言

目前，谷歌拼音输入法扩展API只提供了Lua一种开发语言。Lua是一种体积小巧却功能强大的动态脚本编程语言，广泛用于网络游戏等应用的插件或扩展功能的开发。对于一个有JavaScript语言、VBScript语言或者Python语言开发经验的开发者来说，学习Lua语言并不困难。请参考以下网址获得关于Lua语言的各种信息：

Lua程序设计语言

在输入法环境中，每个用户安装的输入法扩展都是一个独立的Lua语言模块，每个模块在自己的名字空间中运行。也就是说，不同模块间的同名符号不会冲突，但不同模块间也无法相互进行功能调用。用户安装的输入法扩展只能调用API参考一节描述的Lua内置功能函数和ime模块提供的输入法相关函数。

API参考

使用Lua语言编写的输入法扩展程序保存在磁盘上时，推荐使用UTF-8编码的文本文件。文本文件可以包含也可以不包含BOM文件头。没有BOM文件头时，IME缺省认为按UTF-8编码加载。

使用Lua语言编写的输入法扩展程序可以使用以下内置函数。除了ime模块提供的输入法相关函数外，这些函数是标准Lua运行环境的一个子集。因此，在将已有的Lua程序移植到输入法扩展程序之前，请确认程序使用的函数在下表所涵盖的范围内。

以下凡属于Lua标准运行环境的函数，均只给出简要的功能说明。详细用法请参见Lua语言标准函数库的说明。

基本功能函数

- **assert (v [, message])**
- ◦ 断言。如果v的值是非（nil或false），就报告错误。
- **error (message [, level])**
- ◦ 报告错误。
- **ipairs (t)**
- ◦ 迭代器函数。用于对列表中元素的（序号，值）进行迭代。
- **loadstring (string [, chunkname])**
- ◦ 加载并执行字符串。
- **next (table [, index])**
- ◦ 用于遍历列表的每个元素。
- **pairs (t)**
- ◦ 迭代器函数。用于对列表中元素的（键，值）进行迭代。
- **select (index, ...)**
- ◦ 返回变长参数列表中从index之后开始的所有参数。
- **tonumber (e [, base])**
- ◦ 转换为数字。
- **tostring (e)**

- ○ 转换为字符串。
- **type (v)**
- ○ 返回参数的类型名。
- **unpack (list [, i [, j]])**
- ○ 返回列表中的各个元素。

字符串处理函数

- **string.byte (s [, i [, j]])**
- ○ 返回字符内部编码。
- **string.char (…)**
- ○ 返回内部编码对应的字符串。
- **string.find (s, pattern [, init [, plain]])**
- ○ 字符串查找。
- **string.format (formatstring, …)**
- ○ 字符串格式化。
- **string.gmatch (s, pattern)**
- ○ 迭代器函数。用于在字符串中对所有匹配项进行迭代。
- **string.gsub (s, pattern, repl [, n])**
- ○ 字符串全局替换。
- **string.len (s)**
- ○ 返回字符串长度。
- **string.lower (s)**
- ○ 转换为小写。
- **string.match (s, pattern [, init])**
- ○ 字符串匹配。
- **string.rep (s, n)**
- ○ 重复字符串n次。
- **string.reverse (s)**
- ○ 字符串反转。
- **string.sub (s, i [, j])**
- ○ 字符串替换。
- **string.upper (s)**
- ○ 转换为大写。

日期和时间函数

- **os.date ([format [, time]])**
- ○ 返回格式化的日期时间字符串。
- **os.difftime (t2, t1)**
- ○ 返回两个时间相差的秒数。
- **os.time ([table])**
- ○ 返回当前日期，或指定的日期。

数学函数

- **math.abs (x)**
- ○ 绝对值。
- **math.acos (x)**
- ○ 反余弦。

- **math.asin (x)**
- ○ 反正弦。
- **math.atan (x)**
- ○ 反正切。
- **math.ceil (x)**
- ○ 向上取整。
- **math.cos (x)**
- ○ 余弦。
- **math.cosh (x)**
- ○ 双曲余弦。
- **math.deg (x)**
- ○ 弧度转角度。
- **math.exp (x)**
- ○ 计算e^x
- **math.floor (x)**
- ○ 向下取整。
- **math.fmod (x, y)**
- ○ 浮点数取模。
- **math.frexp (x)**
- ○ 返回使x = $m2^e$成立的m和e
- **math.ldexp (m, e)**
- ○ 计算$m2^e$
- **math.log (x)**
- ○ 计算自然对数。
- **math.log10 (x)**
- ○ 计算常用对数。
- **math.max (x, ···)**
- ○ 求最大值。
- **math.min (x, ···)**
- ○ 求最小值。
- **math.modf (x)**
- ○ 返回浮点数的整数和小数部分。
- **math.pi**
- ○ 返回n值。
- **math.pow (x, y)**
- ○ 计算x^y
- **math.rad (x)**
- ○ 角度转弧度。
- **math.random ([m [, n]])**
- ○ 生成伪随机数。
- **math.randomseed (x)**
- ○ 设置随机数种子。
- **math.sin (x)**
- ○ 正弦。
- **math.sinh (x)**

- ◦ 双曲正弦。
- **math.sqrt (x)**
- ◦ 平方根。
- **math.tan (x)**
- ◦ 正切。
- **math.tanh (x)**
- ◦ 双曲正切。

表处理函数

- **table.concat (table [, sep [, i [, j]]])**
- ◦ 连接列表元素。
- **table.insert (table, [pos,] value)**
- ◦ 插入元素。
- **table.maxn (table)**
- ◦ 返回最大整数索引号。
- **table.remove (table [, pos])**
- ◦ 删除元素。
- **table.sort (table [, comp])**
- ◦ 排序。

ime模块提供的输入法相关函数

注册输入法模块相关函数

- **ime.register_command (command_name, lua_function_name, description [, leading [, help]])**
- ◦ 注册命令扩展。详细说明参见注册命令扩展。
- **ime.register_trigger (lua_function_name, description, input_trigger_strings, candidate_trigger_strings)**
- ◦ 注册整合扩展。详细说明参见注册整合扩展。
- **ime.register_converter (lua_function_name, description)**
- ◦ 注册转换器扩展。详细说明参见注册转换器扩展。

输入法信息相关函数

- **ime.get_version ()**
- ◦ 以字符串方式返回当前输入法的版本号。
- **ime.get_last_commit ()**
- ◦ 返回用户通过谷歌拼音输入法输入的上一个字符串。

实用工具函数

- **ime.int_to_hex_string(value [, width])**
- ◦ 将整数值value转换成16进制表示的字符串。可选参数width指定了结果字符串的最小长度，不足最小长度时，高位用"0"补齐。
- **ime.join_string (str_list, sep)**
- ◦ 将str_list列表中的所有字符串连接成一个大字符串，并使用sep作为连接字符，返回结果字符串。
- **ime.parse_mapping (src_string, line_sep, key_value_sep, values_sep)**
- ◦ 字符串src_string是一个用字符串方式表示的<键, 一个或多个值>的映射表，每一行表示一个键与其值的映射，行与行之间的分隔字符是line_sep，每行内键与后续的一个或多个值之间的分隔字符是key_value_sep，多个值之间的分隔字符是values_sep。

ime.parse_mapping解析此格式的字符串，将其转换为Lua语言可直接使用的列表。结果列表的格式为{key1={value1, value2, ...}, key2={value1, value2, ...}, ...}。此工具函数的典型应用实例是：在输入法扩展模块中预置一个字符串形式、易于阅读和编辑的映射表（比如一种汉字编码到对应的一个或多个汉字的映射），然后在扩展模块中用此函数解析该映射表，以便Lua代码快速查询。

- 示例代码：
- _MAPPING_TABLE = [[a 啊b 不,吧c 从,穿,出]]_MAPPING = ime.parse_mapping(_MAPPING_TABLE, "\n", " ", ",")function Lookup(input) if _MAPPING[input] then return _MAPPING[input] else error("Invalid argument") endendime.register_command("lp", "Lookup", "mapping lookup")

- **ime.split_string (str, sep)**
- ◦ 根据分隔字符sep，将字符串str拆分为一组字符串，返回这一组字符串组成的列表。
- **ime.trim_string (str)**
- ◦ 去除字符串str左右两边的空白字符，返回结果字符串。
- **ime.trim_string_left (str)**
- ◦ 去除字符串str左边的空白字符，返回结果字符串。
- **ime.trim_string_right (str)**
- ◦ 去除字符串str右边的空白字符，返回结果字符串。
- **ime.utf8_to_utf16 (str)**
- ◦ 将UTF-8编码的字符串转换为UTF-16编码的字符串，返回的结果字符串以"\0\0"结尾。
- **ime.utf16_to_utf8 (str)**
- ◦ 将UTF-16编码的字符串转换为UTF-8编码的字符串。

标签： Google 输入法API

» 下一篇：JavaScript多线程编程简介