正则表达式作为一种通用的算式，大量用于文本处理逻辑中，其主要好处是速度能够达到极致，而且代码量通常会非常少，本篇主要介绍lua中的正则使用，由于这样的文章网上随处都可以搜到，而且不乏整理的不错的，这里就不想再重新做这件事情。

正则表达式主要处理匹配问题，即通过给定语义通配符串，返回匹配成功的字符串列表。

在匹配的基础上，正则引擎往往提供了字符串的查找、替换、分割等功能，lua支持查找与替换，但不支持分割，分割函数可以在已有接口上进行扩展，附录中引用了lua社区的实现代码。

本篇单纯的简要介绍lua的匹配问题。

Lua正则中的特殊字符（元字符）包括如下几种：

**( ) . % + - \* ?[ ] ^ $**

**转义字符（元字符%）：**

'%' 用作特殊字符的转义字符，'%.' 匹配点；'%%' 匹配字符 '%'，’%’与特定字母结合有特定含义（通配符），如下：

%a: 与任何字母配对  
%c: 与任何控制符配对(例如\n)  
%d: 与任何数字配对  
%l: 与任何小写字母配对  
%p: 与任何标点(punctuation)配对  
%s: 与空白字符配对  
%u: 与任何大写字母配对  
%w: 与任何字母/数字配对  
%x: 与任何十六进制数配对  
%z: 与任何代表0的字符配对

对于上面的%x（其中x代表a,c,d,...,z），lua正则支持其大写形式，大写形式代表非%x，即有%X = ^%x，比如%A表示与任意非字母配对。

**其他通配符（元字符.）：**

.(点): 与任何字符配对

**字符集（元字符[]）：**

出了上面提到的通配符，我们还可以自定义字符集，如[%a%d]就是与字母和数字配对，我们还可以使用’-‘符号来指定一个范围，如[0-7]表示[01234567]，[a-z]表示小写字母，通过字符集，我们可以定义其他复杂的通配符。

**贪婪与吝啬（元字符+、-、？、\*）**

+      匹配前一字符1次或多次  
\*      匹配前一字符0次或多次  
-      匹配前一字符0次或多次  
?      匹配前一字符0次或1次

元字符+和\*是贪婪的，总是进行最长的匹配，而-则是吝啬的，总是进行最短匹配，注意元字符-可以匹配0次。例子：

待匹配的字符串：<font>a</font><font>b</font>

模式串（1）：<font>.+</font>此时将匹配整个字符串，贪婪模式下，正则引擎即使发现了第一个匹配，也不会停止，因此效率相对较低。

模式串（2）：<font>.-</font>此时将依次匹配<font>a</font>、<font>b</font>，最短匹配模式下，一旦正则引擎发现第一个匹配就停止动作，不会继续匹配，

**开头与结尾（元字符^、$）**

^x表示非x，可以理解为正则引擎发现了x字符，那么就可以从这里作为一个匹配的开头，匹配过程中可能希望有多个开头（特别是吝啬模式），这个符号用的频率会比较高，x$表示以x结尾，这里的结尾指的是整个待匹配字符串的最后一个字符，这个符号使用的场合和频率相对会比较少。

**取己所需（元字符()）**

圆括号是一个很好用的元字符，如上<font>a</font>，如果我们只是想让lua正则引擎返回字符a，可以采用模式<font>(.+)</font>，否则它会返回整个字符串。