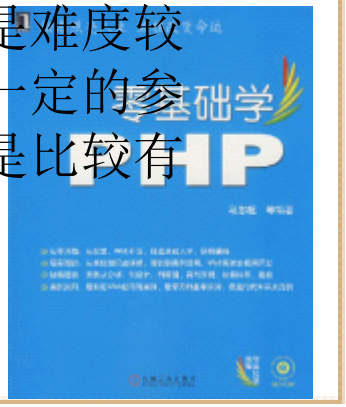
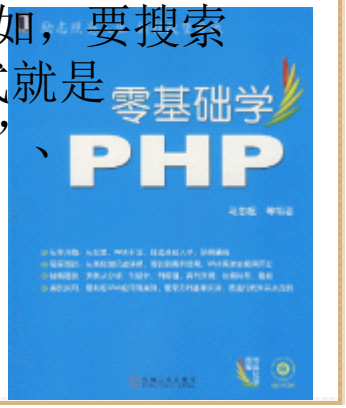
第15章　PHP中的正则表达式

- 正则表达式是在处理文本数据时，一项重要而复杂的技术。许多语言，包括PERL、PHP、Python、JavaScript都支持使用正则表达式处理文本，一些文本编辑器用正则表达式实现高级别的“搜索/替换”功能。正则表达式已经在很多软件中得到广泛的应用，包括Linux、Unix等操作系统，在PHP、Java等开发环境，以及很多的应用软件中，都可以看到正则表达式的影子。
- 使用正则表达式的目的就是，可以通过简单的办法来实现强大的功能。为了简单有效而又不失强大，造成了正则表达式规则的复杂，构建正确有效的正则表达式更是难度较大，所以需要付出一些努力才行。入门之后通过一定的参考和大量实践，在开发实践中使用正则表达式还是比较有效并且有趣的。



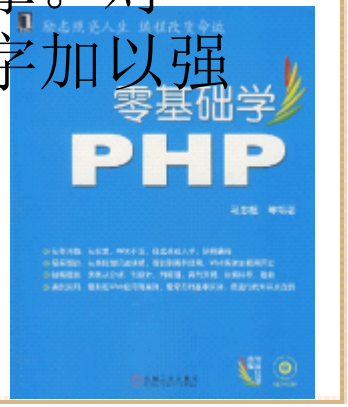
15.1　什么是正则表达式

- 如果原来没有使用过正则表达式，那么读者可能对这个术语会完全不熟悉，对读者来说这的确是一个陌生的概念。其实，平时在使用计算机的时候多少都会碰到正则表达式的影子。例如，要在计算机系统中查找某个文件，碰巧忘记了文件名，但知道该文件的类型，即知道该文件的后缀名，热纾　　乙桓璧计　　敲纯赡芑嵬ü*.png这样的字符来帮助查找，其中字符*就代表了一个或多个字符。计算机通过这样的字符组合，会将系统中所有以.png为后缀名的文件列出来，如：m.png、flag.png、river.png、mydog.png等，以便用户找到需要的图片文件。
- 事实上，可以将这个例子中的*.png看做是一个正则表达式，利用这个表达式，计算机匹配出符合条件的多个文件。又如，要搜索一个包含字符“cat”的字符串，搜索用的正则表达式就是“cat”。如果搜索对大小写不敏感，单词“catalog”、“Catherine”、“sophisticated”都可以匹配。



15.2　正则表达式的语法

- 仅从概念上了解正则表达式是没有实际作用的，对于掌握、理解直至熟练应用正则表达式是还需要从正则表达式的构成等方面深入了解正则表达式。这一节将介绍正则表达式的基本语法，以及简述如何通过这些语法构建一个正则表达式。这节涉及很多符号以及这些符号的使用规则，有的符号在正则表达式不同的位置所表示的含义完全不同，所以读者需要仔细阅读，用心揣摩。对一些需要重点理解的地方，笔者使用粗体字加以强调，读者需对这些讲述认真理解。

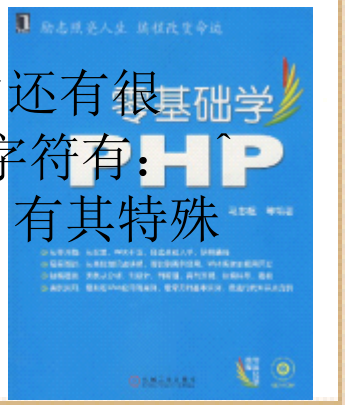


15.2.1 模式

- 模式是正则表达式最基本的元素，它们是一组 **描述字符串特征**的字符。模式可以很简单，由普通的字符串组成，也可以非常复杂。模式往往用一些特殊的字符表示某个范围内的字符、字符重复出现次数、字符是否应该出现等。例如下面的示例。
- ^once
- 以上^once就是一个模式，这个模式包含一个特殊的字符^，该符号表示该模式 **只匹配**那些以once**开头**的字符串。例如该模式与字符串“once you begin, you must continue”匹配，与“all at once she lost her temper”不匹配。正如“^”符号表示开头一样，“$”符号用来匹配那些以给定模式结尾的字符串。例如下面的示例。
- PHP$

15.2.2　元字符

- 到目前为止，读者已经了解到，正则表达式是定义了字符串匹配规则的字符串，该字符串是由普通字符（例如字符 a 到z）以及特殊字符组成的文字模式。普通字符通常是 包括所有的大写和小写字母字符、所有数字、所有标点符号以及一些符号。 正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。例如，正则表达式 Mac*所表达的含义是，所有包含Mac并且Mac后跟零个或多个字符的字符串。Mack、Macintosh、以及 Mac 本身都是符合这个正则表达式所定的规则的。 Mac*中的星号*有其特殊含义，它表示一个或多个字符。
- 类似星号*这样有特殊含义的字符在正则表达式中还有很多，这样的字符叫做元字符。正则表达式中的元字符有：^ $ [ ] \ . | ? * + ( )。这些字符被保留下来，有其特殊用途。下面就分别介绍这些元字符的用途。



15.2.3　转义字符

- 正则表达式中的转义，除了需要对元字符转义之外，还有一些非打印的字符在匹配时需要转义。

非打印字符的转义含义

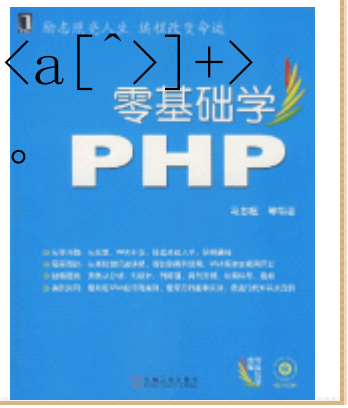
字符	含义描述
\n	匹配一个换行符。等价于\x0a和\cJ。
\r	匹配一个回车符。等价于\x0d和\cM。
\s（小写）	匹配任何空白字符，包括空格、制表符、换页符等等。等价于[\f\n\r\t\v]。
\S（大写）	匹配任何非空白字符。等价于[^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于\x09和\cI。
\v	匹配一个垂直制表符。等价于\x0b和\cK。
\f	匹配一个换页符。等价于\x0c和\cL。
\cx	匹配由x指明的控制字符。例如，\cM匹配一个Control-M或回车符。x的值必须是A-Z或a-z之一。否则，将c视为一个原义的 'c'字符，即字符c本身。

15.2.4　字符类

- PHP的正则表达式有一些内置的通用字符类，可以在正则表达式中直接使用这些字符类，完成对各种字符的匹配，这种字符类的用法相对简单一些。PHP正则表达式通用的字符类包括以下所示内容。
- [[:alpha:]]：表示匹配任何字母。
- [[:digit:]]：表示匹配任何数字。
- [[:alnum:]]：表示匹配任何字母和数字。
- [[:space:]]：表示匹配任何白字符。
- [[:upper:]]：表示匹配任何大写字母。
- [[:lower:]]：表示匹配任何小写字母。
- [[:punct:]]：表示匹配任何标点符号。
- [[:xdigit:]]：表示匹配任何16进制的数字，相当于[0-9a-fA-F]。
- [[:blank:]]：表示空格和TAB，等价于[\t]。
- [[:cntrl:]]：表示匹配所有ASCII 0到31之间的控制符。
- [[:graph:]]：表示匹配所有的可打印字符，等价于：[^ \t\n\r\f\v]。
- [[:print:]]：表示匹配所有的可打印字符和空格，等价于：[^\t\n\r\f\v]。

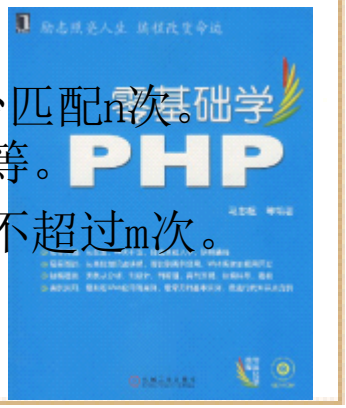
15.2.5　反义

- PHP正则表达式反义匹配主要有以下几种。
- \W匹配任意不是子母，数字，下划线或汉字的字符。
- \S匹配任意不是空白符的字符。
- \D匹配任意非数字的字符。
- \B匹配不是单词开头或结束的位置，即一个词语的边界,位于\w和\W之间的位置。
- 例如，\S+匹配不包含空白符的字符串，<a[^>]+>匹配用尖括号括起来的以a开头的字符串。



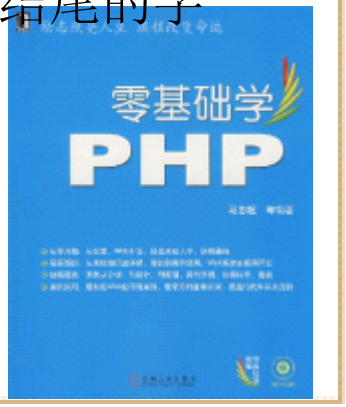
15.2.6　数量匹配限定符

- 数量限定符用来指定正则表达式中，一个给定组合必须要出现多少次才能满足匹配。在15.2.2小节，介绍元字符时，对一部分匹配限定符有所介绍，它们是：*、+、?。这小节将要介绍另外一种匹配限定符，使用这一类限定符也可以完成**对字符出现次数的匹配**。
- 使用符号“{}”可以确定其左边（即前面）的内容重复出现的次数，在{}内指定字符出现的次数，通常有3中表示法，它们如下所示。
- {n}：表示匹配该限定符左边字符n次。例如a{3}，该模式表示匹配连续出现3个a的字符串，它可以匹配aaa、cacaaad、aacoaaao等。
- {n,}：表示匹配该限定符左边字符n次或多次，即至少匹配n次。例如a{3,}，匹配aaa、aaab、caaaaa等，但不匹配aa等。
- {n,m}：表示匹配该限定符左边字符至少n次，但最多不超过m次。例如a{1,3}b，匹配ab、aab、aaab，但不匹配aaaab。



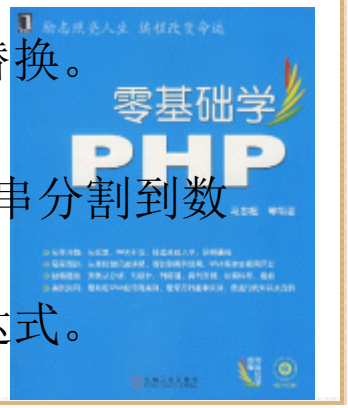
15.2.7　正则表达式构建实例

- 本节将综合以上几小节知识，通过对一些正则表达式的构成分析，来进一步理解以上几小节学习到的知识等。另外，举一些实例，来学习如何创建一个正则表达式。下面先分析几个比较简单的正则表达式。
- ab*：和ab{0,}同义，匹配以a开头，后面可以接0个或者N个b组成的字符串，如a、ab、abbb 等。
- ab+：和ab{1,}同义，但最少要有1个b存在，如ab、abbb等。
- ab?：和ab{0,1}同义，可以没有或者只有1个b，如a、ab。
- a?b+$，匹配以1个或者0个a再加上1个以上的b为结尾的字符串，如ab、abb等。



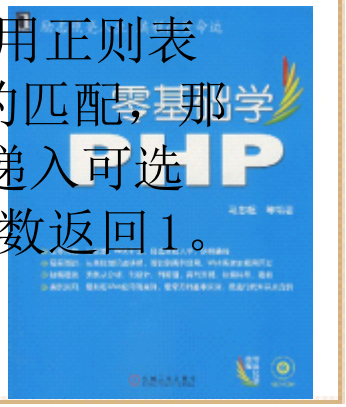
15.3　PHP中的POSIX 扩展正则表达式函数

- 通过前面几章的学习，读者已经认识到，正则表达式可用于复杂字符串的处理。仅仅了解什么是正则表达式和如何创建一个正则表达式，在实际应用中还是无法完成具体工作的。和其他解决问题的办法一样，PHP提供函数完成对正则表达式的支持。PHP有两大类函数支持正则表达式，一类是POSIX扩展函数，另一类是PERL兼容的正则表达式函数。本小结先介绍POSIX扩展正则表达式函数，这些函数如下所示。
- 函数ereg()，字符串的正则匹配函数。
- 函数ereg_replace()，不区分大小写的正则表达式替换。
- 函数eregi()，不区分大小写的正则表达式匹配。
- 函数eregi_replace()，不区分大小写的正则表达式替换。
- 函数split()，用正则表达式将字符串分割到数组中。
- 函数spliti()，用正则表达式不分字母大小写将字符串分割到数组中。
- 函数sql_regcase()，产生用于不区分大小的正则表达式。



15.3.1　正则表达式匹配函数

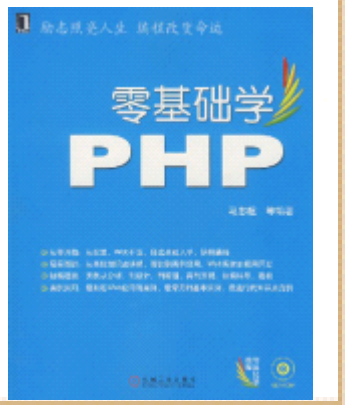
- 在PHP中完成对正则表达式是否匹配判断的函数是 ereg()，该函数的原型如下所示。
- int ereg (string $pattern, string $string [, array &regs])
- 该函数以区分大小写的方式，按参数 $pattern（该参数是一个正则表达式）的规则来匹配字符串参数 $string。匹配结果所返回的值放在第3个可选数组参数regs之中，regs[0]内容就是原字符串string，regs[1]为第一个合乎规则的字符串，regs[2] 就是第二个合乎规则的字符串，依次类推。若调用该函数是，省略参数regs，则只是单纯使用正则表达式做匹配，如果在string中找到pattern模式的匹配，那么该函数返回所匹配字符串的长度，如果没有传递入可选参数regs或者所匹配的字符串长度为 0，则本函数返回1。如果没有找到匹配或出错，该函数返回 FALSE。



15.3.2　替换匹配字符串的函数

- 替换匹配字符串的正则表达式函数有两个，一个是函数 ereg_replace()，另一个是函数eregi_replace()，它们之间的区别是，是否忽略匹配字母的大小写。下面分别加以介绍，函数ereg_replace()的语法如下所示。
- string ereg_replace (string $pattern, string $replacement, string $string)
- 该函数在在参数$string中扫描与参数$pattern匹配的部分，并将其用参数$replacement替换。该函数返回替换后的字符串。如果在参数 $string 中未找到匹配正则表达式的项，则参数 $string将原样返回。注意，该函数在匹配字符串时，是区分大小写的。

使用函数 eregi_replace 进行字符串替换

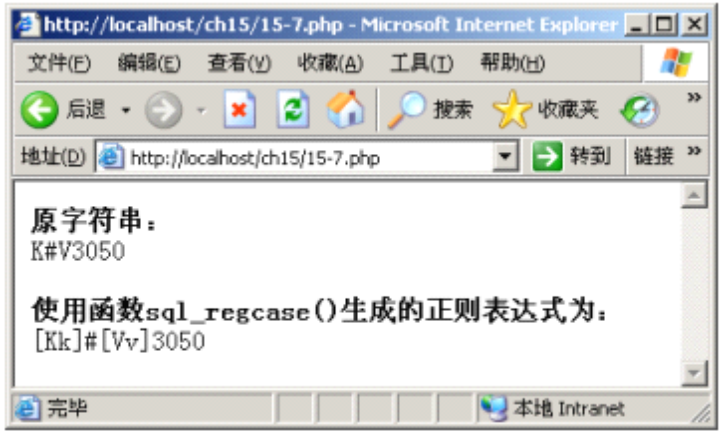


15.3.3 根据正则表达式分割字符串的函数

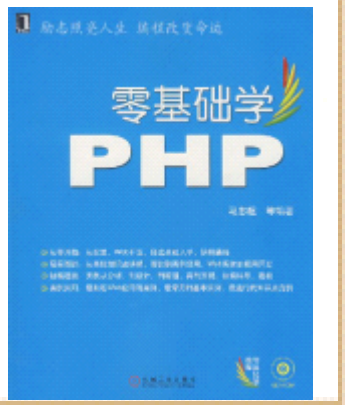
- 函数split()通过匹配正则表达式作为分割符，将字符串分割出各个字串，存入数组中。该函数的语法如下所示。
- array split (string $pattern, string $string [, int $limit])
- 该函数的返回值是一个数组，其每个单元都是参数$string经正则表达式$pattern做为分割符分出的字串。该函数第3个参数是可选参数，如果指定了参数$limit，则返回数组最多包含limit个单元，其中最后一个单元包含参数字符串$string剩余的所有部分。如果执行出现错误，函数split()返回FALSE。

15.3.4　生成正则表达式的函数

- 函数sql_regcase()可以产生用于不区分大小正则表达式，该函数语法如下所示。
- string sql_regcase (string $string)
- 该函数返回与参数$string相匹配的正则表达式，该表达式以不区分大小写的形式返回。

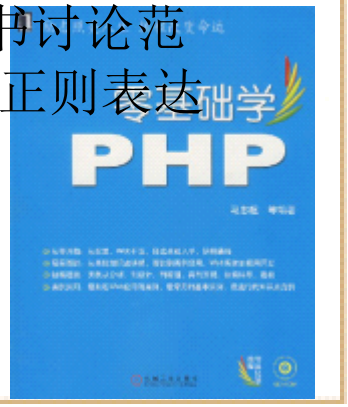


使用函数 sql_regcase 生成正则表达式



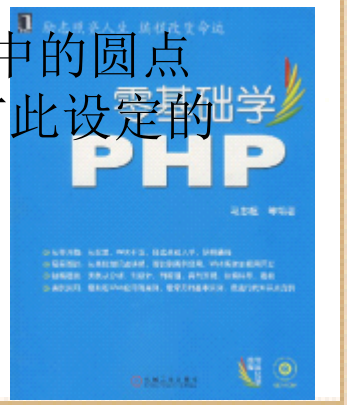
15.4　PHP中的PERL兼容正则表达式函数

- 对PERL语言有所了解的读者，一定对 PERL中正则表达式的强大功能印象深刻。PERL最初就是设计为用来处理文本文件的一中语言，至今， PERL语言在处理文本方面的强大功能仍是无可比拟，尤其在维护、分析系统数据方面， PREL语言扮演着不可替代的角色。
- 正是基于此，PHP中除了可以使用POSIX函数支持正则表达式之外，还可以使用PERL兼容的正则表达式函数。从PHP4.0开始，包含了一个PERL兼容的正则表达式（PCRE）库，和正常regex库一起与PHP绑定。另外，PCRE和PERL的正则表达式之间有一些细微差别，但这并不在本书讨论范围之内。本节就为读者介绍如何使用 PERL兼容的正则表达式函数。



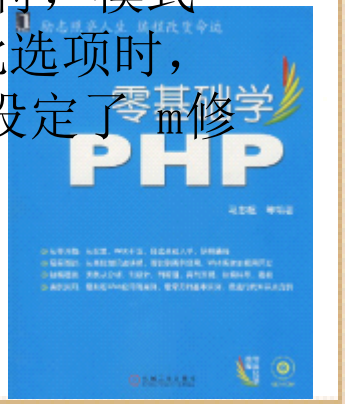
15.4.1　PERL兼容正则表达式中的修正符

- 在正式介绍PERL兼容正则表达式函数之前，先来了解一下PERL兼容正则表达式中可能使用的修正符。所谓修正符，是指在正则表达式最后诸如/si之类的修正说明。这些修正符如下所示，括号中的名称是这些修正符的内部 PCRE名，对于这个名称，读者可以不必过于关注。
- i（PCRE_CASELESS）：匹配时忽略大小写。
- m（PCRE_MULTILINE）：当设定了此修正符，行起始(^)和行结束($)除了匹配整个字符串开头和结束外，还分别匹配其中的换行符(\n)的之后和之前。
- s（PCRE_DOTALL）：如果设定了此修正符，模式中的圆点元字符（.）匹配所有的字符，包括换行符。没有此设定的话，则不包括换行符。



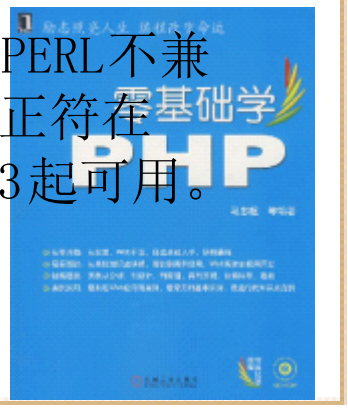
15.4.1　PERL兼容正则表达式中的修正符

- x（PCRE_EXTENDED）：如果设定了此修正符，模式中的空白字符除了被转义的或在字符类中的以外完全被忽略。
- e：如果设定了此修正符， preg_replace()在替换字符串中对逆向引用作正常的替换，将其作为 PHP 代码求值，并用其结果来替换所搜索的字符串。 只有preg_replace()使用此修正符，其它 PCRE 函数将忽略之。
- A（PCRE_ANCHORED）：如果设定了此修正符，模式被强制为“anchored”，即强制仅从目标字符串的开头开始匹配。
- D（PCRE_DOLLAR_ENDONLY）：如果设定了此修正符，模式中的行结束($)仅匹配目标字符串的结尾。没有此选项时，如果最后一个字符是换行符，也会被匹配。如果设定了m修正符则忽略此选项。



15.4.1　PERL兼容正则表达式中的修正符

- S：当一个模式将被使用若干次时，为加速匹配起见值得先对其进行分析。如果设定了此修正符则会进行额外的分析。目前，分析一个模式仅对没有单一固定起始字符的 non-anchored模式有用。
- U（PCRE_UNGREEDY）：使“?”的默认匹配成为贪婪状态的。
- X（PCRE_EXTRA）：模式中的任何反斜线后面跟上一个没有特殊意义的字母导致一个错误，从而保留此组合以备将来扩充。默认情况下，一个反斜线后面跟一个没有特殊意义的字母被当成该字母本身。
- u（PCRE_UTF8）：此修正符启用了一个PCRE中与PERL不兼容的额外功能。模式字符串被当成 UTF-8。本修正符在UNIX下自PHP4.1.0起可用，在Win32下自PHP4.2.3起可用。自PHP 4.3.5起开始检查模式的UTF-8合法性。

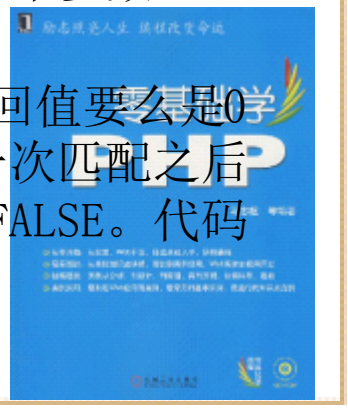


15.4.2　返回与模式匹配的数组单元的正则表达式函数

- 从这小节开始PERL兼容的正则表达式函数，这类函数中所使用的模式极其类似PERL语言。表达式应被包含在定界符中，如斜线（/）。这就是说，正则表达式作为参数传入这些函数时，需要将正则表达式限定在 //之内。例如：/<\/\w+>/，而/href='(.*)'就不是一个合法的PERL兼容正则表达式，因为它缺少结束界定符 /。任何不是字母、数字或左斜杠（\）的字符都可以作为定界符。如果作为定界符的字符必须被用在表达式本身中，则需要用反斜线转义。
- 第一个要介绍的是函数preg_grep()，该函数返回与模式匹配的数组元素。函数preg_grep()的语法如下所示。
- array preg_grep (string $pattern, array $input [, int $flag])

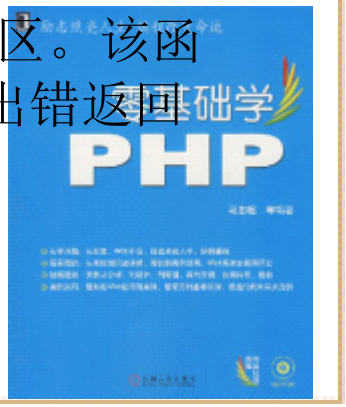
15.4.3　进行正则表达式匹配的函数

- PERL兼容的正则表达式函数preg_match()，用来完成对正则表达式的匹配。该函数的语法如下所示。
- int preg_match (string $pattern, string $subject [, array $matches [, int $flag]])
- 该函数在参数字符串subject中搜索与参数$pattern给出的正则表达式相匹配的内容。该函数的第3个参数是可选参数，该参数是一个数组，如果提供了第3个参数$matches，则其会被搜索的结果所填充。$matches[0]包含与整个模式匹配的文本，$matches[1]将包含与第一个捕获的括号中的子模式所匹配的文本，以此类推。函数perg_match()还有第4个参数，也是可选的，对这个参数，这里不再详细叙述。
- 函数perg_match()返回对$pattern的匹配次数，该返回值要么是0次（没有匹配）或1次，因为函数preg_match()在第一次匹配之后将停止继续匹配。如果出错，函数preg_match()返回FALSE。代码15-9演示了该函数的用法。



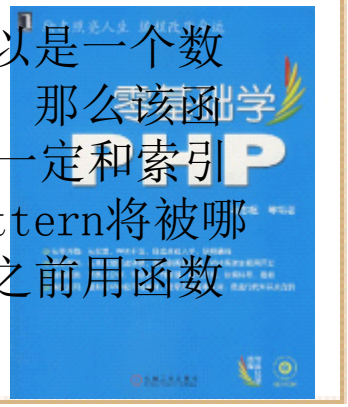
15.4.4　进行全局正则表达式匹配的函数

- 和函数preg_match()极为类似的一个函数是preg_match_all()，该函数进行全局正则表达式匹配。它的语法如下所示。
- int preg_match_all (string $pattern, string $subject, array $matches [, int $flag])
- 该函数在参数$subject中搜索**所有**与参数$pattern所给出的正则表达式匹配的内容，并将结果以参数 $flag指定的顺序放到参数数组$matches中。该函数搜索到第一个匹配项之后，会继续进行匹配搜索，接下来的搜索从上一个匹配项末尾开始，这也是该函数preg_match()的一个区。该函数返回整个模式匹配的次数（可能为零），如果出错返回FALSE。



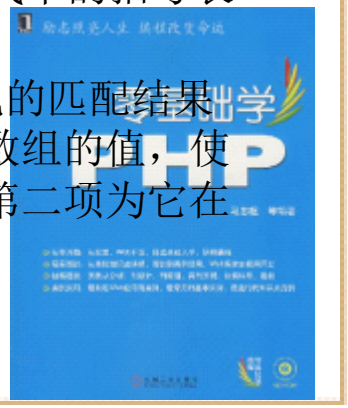
15.4.5　执行正则表达式的搜索和替换的函数

- 函数preg_replace()可以完成对正则表达式的搜索和替换，该函数的语法如下所示。
- mixed preg_replace (mixed $pattern, mixed $replacement, mixed $subject [, int $limit])
- 该函数在参数$subject中搜索参数$pattern模式的匹配项，并替换为参数$replacement所指定的值。该函数的第4个参数是可选的，如果指定了参数$limit，那么该函数仅替换limit个匹配，如果省略参数$limit或者其值为-1，则所有的匹配项都会被替换。
- 函数preg_replace()如果搜索到匹配项，则会返回被替换后的$subject，否则返回原来不变的参数$subject。函数preg_replace()的每个参数（除了参数$limit）都可以是一个数组。如果参数$pattern和参数$replacement都是数组，那么该函数将以其键名在数组中出现的顺序来进行处理。这不一定和索引的数字顺序相同。所以，如果使用索引来标识哪个pattern将被哪个replacement来替换，应该在调用 preg_replace()之前用函数ksort()对数组进行排序。



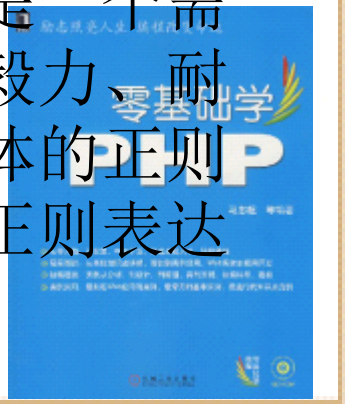
15.4.6　用正则表达式分割字符串的函数

- 和函数split()一样，兼容PERL正则表达式的函数preg_split()也可以通过一个正则表达式，来完成对字符串的分割。该函数的语法如下所示。
- array preg_split (string $pattern, string $subject [,int $limit [, int $flag]])
- 该函数返回一个数组，这个数组包含参数$subject中按与参数$pattern匹配的边界所分割的子串。如果指定了可选的第3个参数$limit，则最多返回limit个子串，如果limit是-1，则表示没有限制，可以用来继续指定可选参数$flag。该函数的第4个参数$flag是可选的，它有如下取值。
- PREG_SPLIT_NO_EMPTY：如果设定了本标记，则preg_split()只返回非空的部分。
- PREG_SPLIT_DELIM_CAPTURE：如果设定了本标记，定界符模式中的括号表达式也会被捕获并返回。
- PREG_SPLIT_OFFSET_CAPTURE：如果设定本标记，对每个出现的匹配结果也同时返回其附属的字符串偏移量。注意，这改变了返回的数组的值，使其中的每个单元也是一个数组，其中第一项为匹配字符串，第二项为它在$subject中的偏移量。



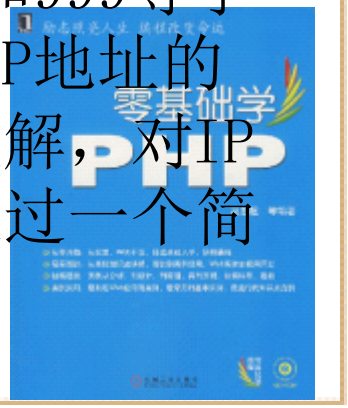
15.5　几例常见正则表达式分析

- 前面4节讲述了正则表达式的基本概念、正则表达式常用字符分类、PHP中POSIX 扩展正则表达式函数及兼容PERL的正则表达式函数。正则表达式这些看似“怪异”的语法，有时的确会让读者感到困惑，尤其对初学者还说，那类似天书一般的正则表达式代码，更是让人望而却步、无所适从。
- 无论怎样，要想掌握并且熟练使用正则表达式，肯定是需要一个渐进过程的。这个过程是一个需要大量实践和积累的过程，还需要一些毅力、耐力和信心。本节就通过几个完整而且具体的正则表达式实例，进一步学习、理解和应用正则表达式。



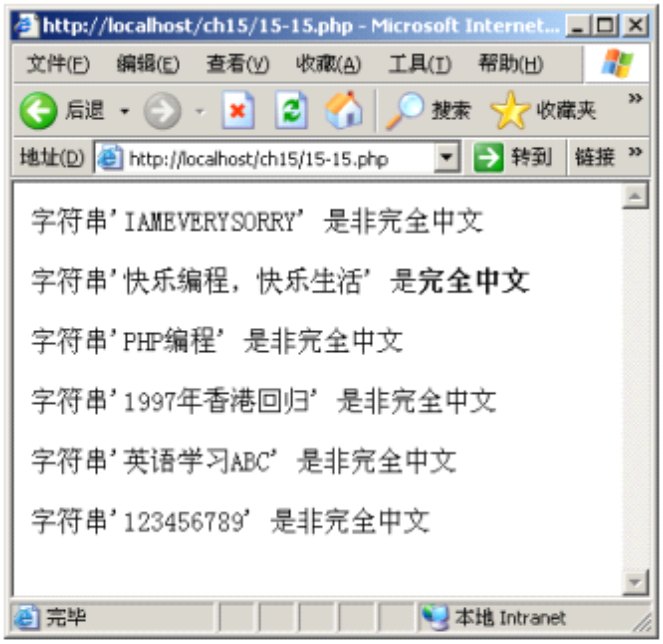
15.5.1　检查IP地址的正则表达式

- IP地址由句点“.”分割的几部分数字字符串组成，例如192.0.170.10。其通用形式是xxx.xxx.xxx.xxx，这里的xxx整数，范围是0到255。因此IP地址中由“.”分割的每个数字串，至少有一个数字、至多有3个数字。既然xxx都是数字，那么首先考虑到的正则表达式[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}。因为[0-9]{1,3}完全可能匹配256、515和999等字符串，所以这个正则表达式并不能保证IP地址的每个数字部分都不超过255。为了便于理解，对IP地址数字部分是否超过255的判断，将通过一个简单的比较表达式来完成。

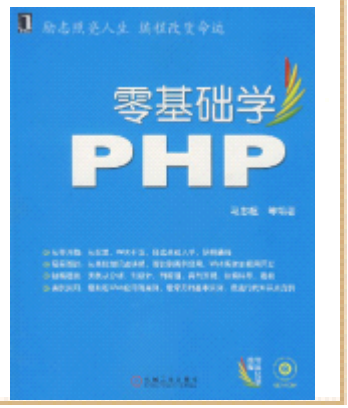


15.5.2　检查中文字符的正则表达式

- 中文在ASCII码中有一定的范围，这个范围是从0xa1到0xff，所以可以用正则表达式0xa1-0xff来表示中文。

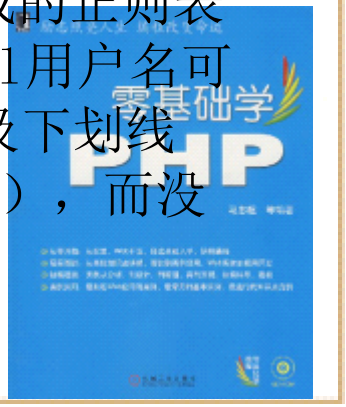


使用正则表达式匹配中文字符串



15.5.3　检查Email地址的正则表达式

- 这一小节介绍如何使用正则表达式来检查一个 Email地址是否是合法的格式。Email地址一般由3部分构成，它们是：
- 用户名，即符号@左边的所有字符。
- 符号@。
- 服务器域名，即符号@右边的所有字符，通常这些字符应该组成一个合法的域名，如 php.net等。
- 通常不同的邮箱提供商对 Email的用户名有不同的限制，如用户名长度不能低于6位字符，用户名只能有数字和字母组成，不能有特殊符号，如下划线 _等。这节要完成的正则表达式，将不过多考虑这些细节，本例将假设 Email用户名可以含有大小写字母、阿拉伯数字、减号（ -）以及下划线（_）。域名的规则也类似，但是有英文句点（ .），而没有下划线。

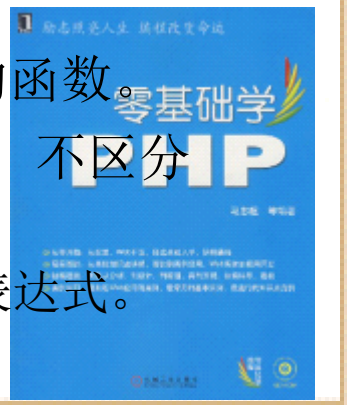


15.5.4　检查URL地址的正则表达式

- 这小节将讲述如何创建匹配URL地址的正则表达式。一般URL地址由以下几个部分构成。
- 协议名，通常是http。
- ://。
- 域名部分。
- 路径文件部分，如/abc/123.html。
- 在15.4.3小节已经使用过域名匹配的正则表达式，这小节只需完成其他部分的正则表达式即可。对于 ://在正则表达式中需要对字符/转义，即\/\/。另外对于整个http://部分，如果URL中不含这部分而URL的其他部分是正确的，那么这样格式的URL也应该算是合法的URL，即http://www.php.net和www.php.net都算是合法的URL。因此，该正则表达式的开头部分是：^(http:\/\/)?。

15.6 小结

- 本章讲述的主要内容有，正则表达式的基本概念、正则表达式的基本语法、PHP中正则表达式函数、PHP中兼容PERL正则表达式的函数、正则表达式实例应用。其中讲述到的PHP正则表达式函数如下所示。
- ereg()，正则表达式匹配函数。
- eregi()，不区分字母大小写的正则表达式函数。
- ereg_replace()，正则表达式替换函数。
- eregi_replace()，不区分字母大小写的正则表达式替换函数。
- split()，用正则表达式将字符串分割到数组中的函数。
- spliti()，用正则表达式将字符串分割到数组中，不区分字母大小写。
- sql_regcase()，产生用于不区分大小的匹配的正则表达式。



15.6 小结

- 本章介绍到的PERL兼容正则表达式函数如下所示。
- preg_grep()，该函数返回与模式匹配的数组单元的。
- preg_match()，正则表达式匹配函数。
- preg_match_all()，全局正则表达式匹配函数。
- preg_replace()，执行正则表达式的搜索和替换的函数。
- preg_split()，用正则表达式分割字符串的函数。