# 简介

**$reg就是筛子**

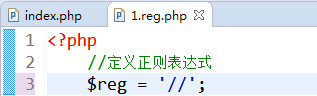
**$str就是沙子**

**Preg\_match参数 （筛子，沙子，筛完之后保存的位置）**

# 规则

## 定界符

默认采用 / / 表示正则表达式的语法



## 元字符

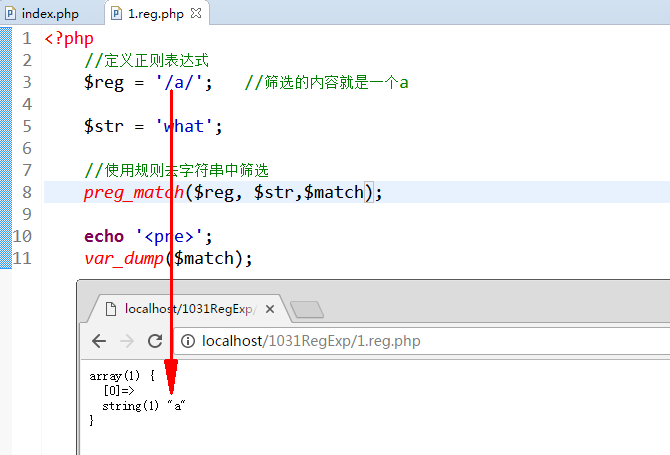
### 直接

元字符，就是使用一些字符（符号），来表示我们筛选的内容

例如： $reg = ‘/a/’; 表示筛选的内容就是a , $reg = ‘/\d/’;表示筛选的是数字

* 一个字符本身就表示一个字符

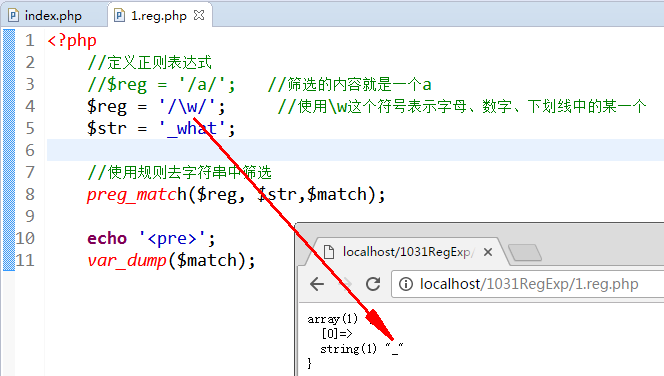
例如：a表示筛选的是a，\_表示筛选的是下划线



### \w

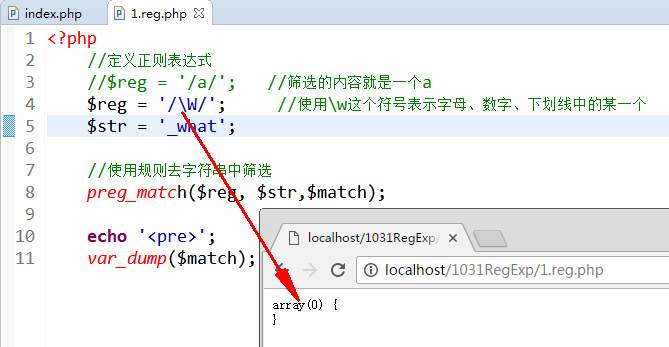
使用转义符号表示某个范围中的一个字符

例如：\w 就表示筛选的内容是字母、数字、下划线中的某一个



### \W

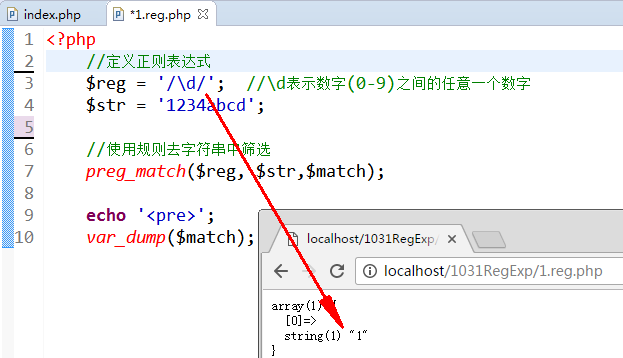
\W：取反，除了字母、数字、下划线中的字符之外的任意一个字符



### \d\D

\d 表示数字中的某一个（0-9之间的任意一个数字）

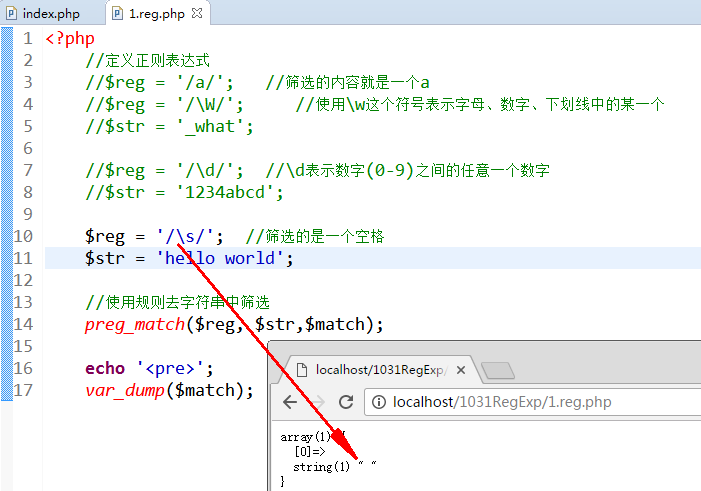
\D 表示除了数字之外的任意一个字符



### \s\S

\s 表示一个空格

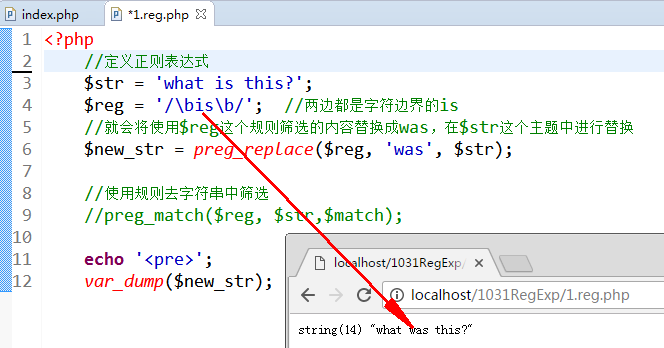
\S 表示除了空格之外的任意一个字符



### \b\B

\b 该字符表示一个字符边界 boundary

\B 表示\b的相反，除了 字符边界之外的任意一个字符



### []

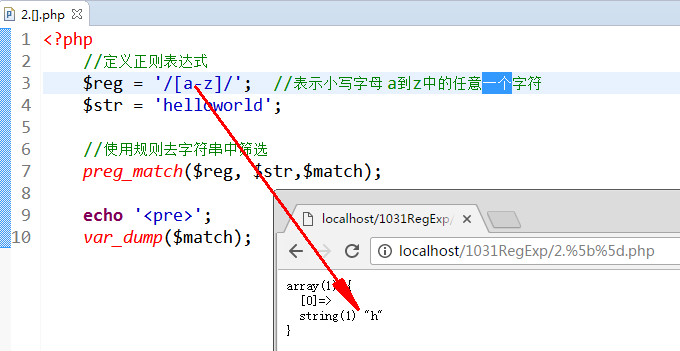
表示集合中的某一个字符，[]表示一个集合

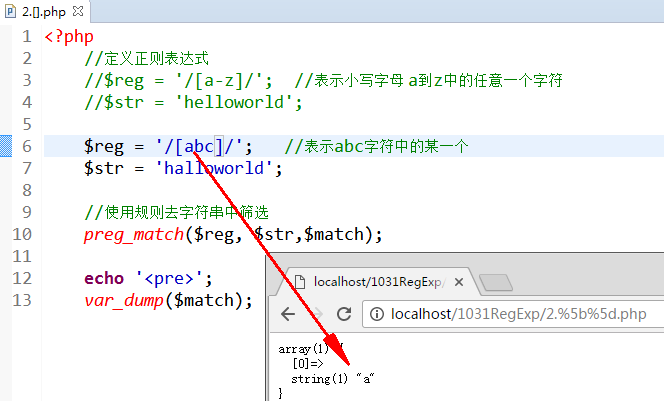
[a-z] ，表示的是小写字母a-z之间的任意一个字符

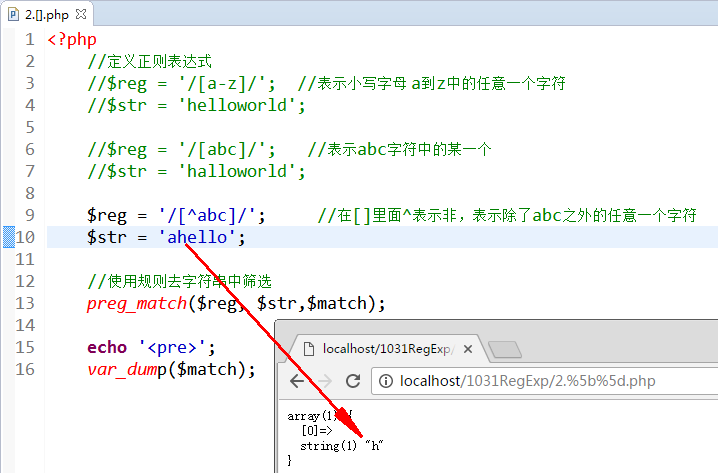
[abc] ，表示abc中的某一个字符

### ^

[^abc] ，[]里面的 ^ 表示非，除了abc之外的任意一个字符



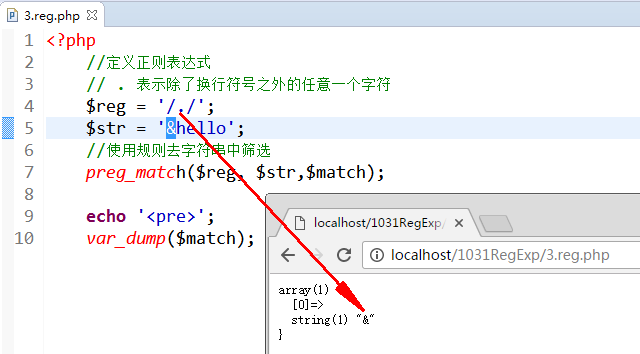




### .

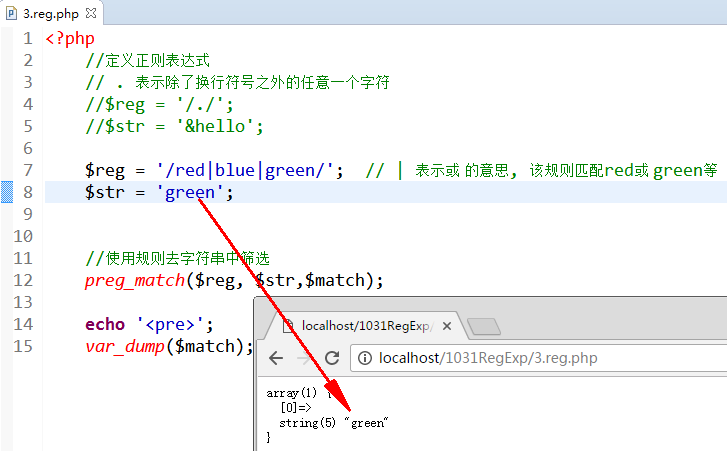
* 特殊符号，表示范围中的某一个

. 表示除了换行符号之外的任意一个字符

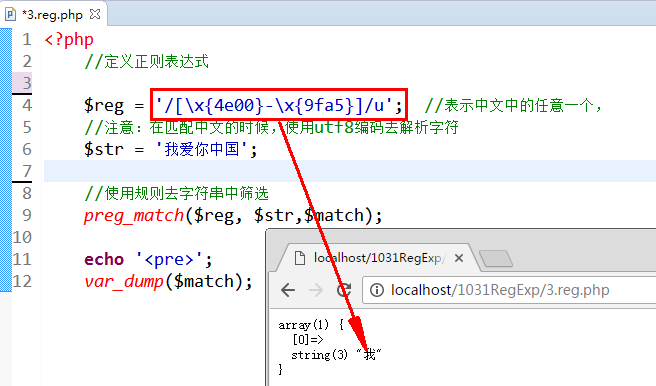


### |

| 或者 red|green|blue，表示红或者绿或者蓝



[\x{4e00}-\x{9fa5}]表示中文的任意一个



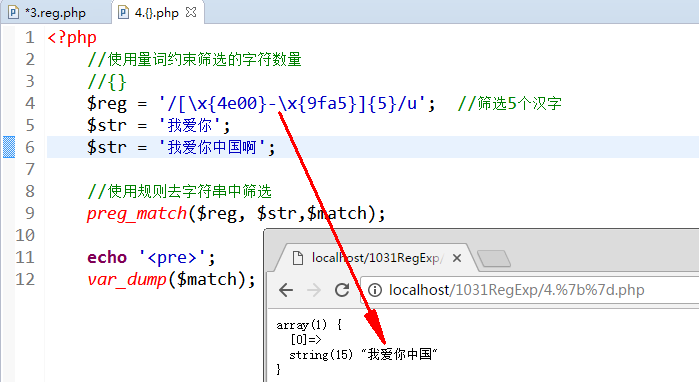
## 量词

### {number}

上面我们筛选的都是一个字符，接下来我们通过量词约束筛选的数量

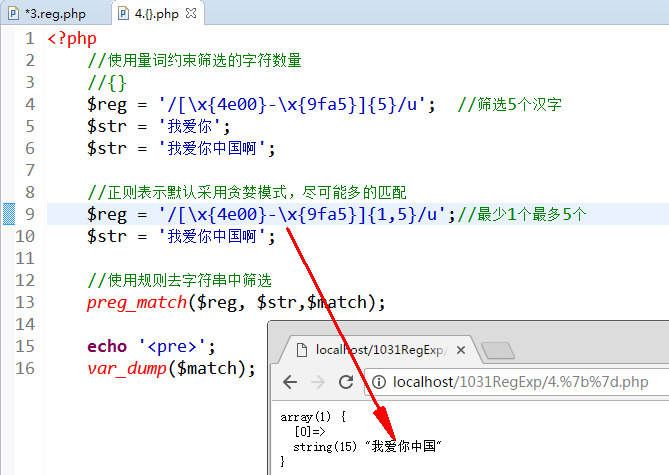
使用{}来约束筛选的数量

{number},筛选固定的number个数量



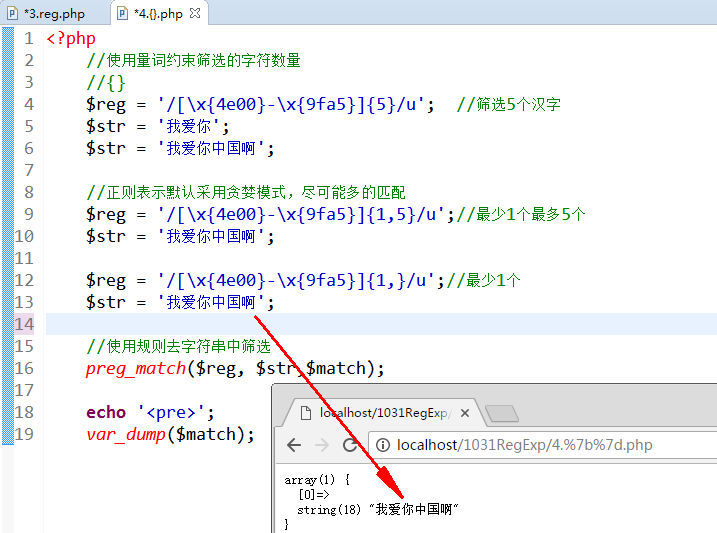
### {min,max}

表示最少min个，最多max个



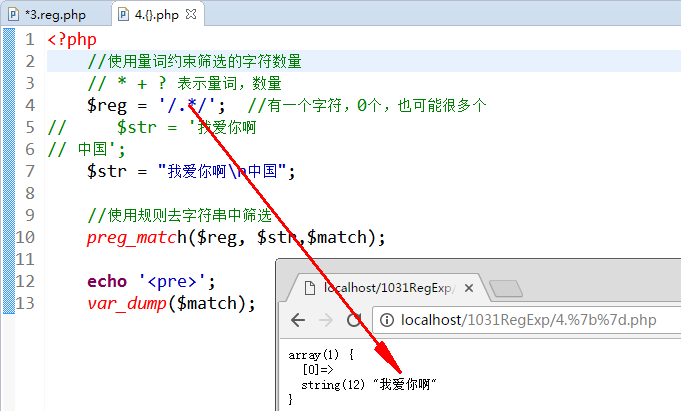
### {min,}

最少min个



### \*

* \* ：任意的数量，可能0个，也可能有更多的数量



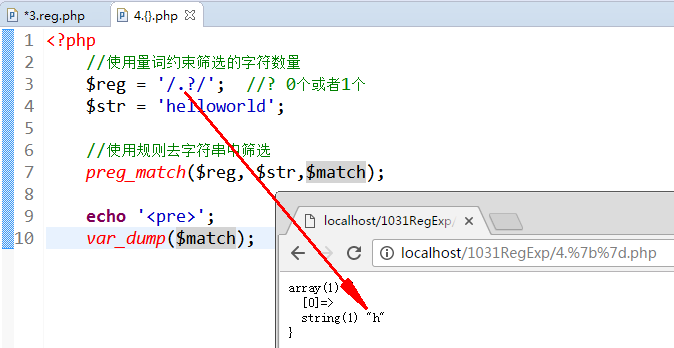
### +

* + ：至少有一个数量



### ?

* ？ ：可能有0个，或1个



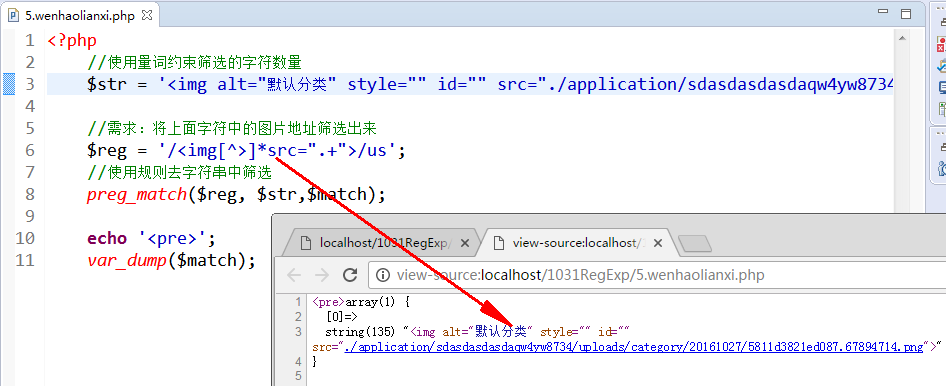
### 贪婪模式懒惰模式

补充：

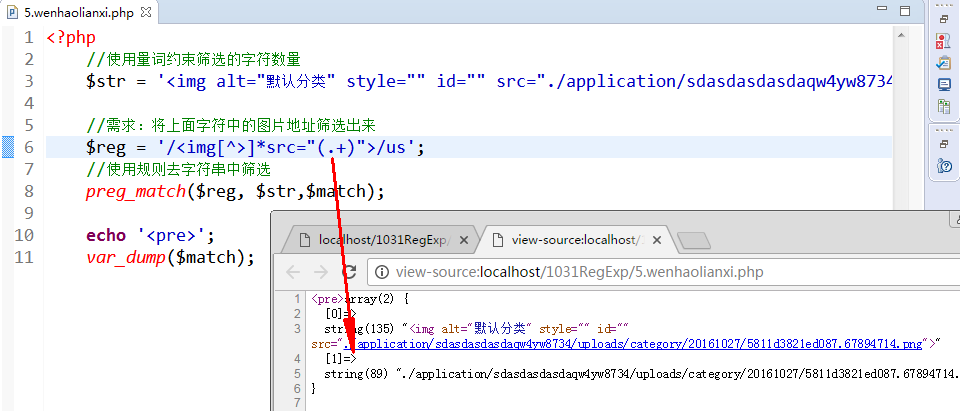
如果将？放在元字符后面，表示约束的是元字符的数量为0个或1个

如果将？放在量词的后面，表示将贪婪模式修改为懒惰模式

1. 先筛选图片地址

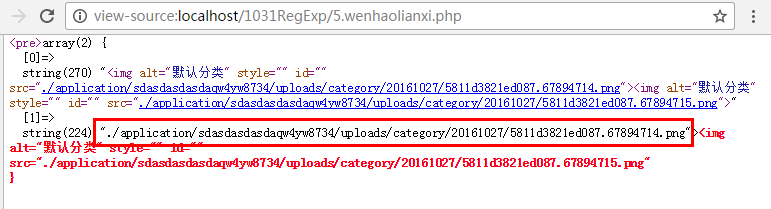


1. 如何筛选出我们指定的数据，例如，上面的案例中，我们只需要图片地址，src属性值，需要使用正则表达式的分组实现，将需要的内容使用（）小括号包起来，返回数据的时候，第一个小括号的内容，保存到数组下标为1的，第二个小括号匹配的内容，保存到数组的下标为2的元素中，以此类推



（3）如果有多张图片怎么筛选？

|  |
| --- |
| //使用量词约束筛选的字符数量  $str = '<img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894714.png"><img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894715.png">';    //需求：将上面字符中的图片地址筛选出来（//u斜线外面的u表示采用万能点模式）  $reg = '/<img[^>]\*src="(.+)">/us';  //使用规则去字符串中筛选  *preg\_match*($reg, $str,$match);    **echo** '<pre>';  *var\_dump*($match); |



如果有多张图片的时候，匹配到14.png并没有结束，继续往后匹配了，因为分组里面写的是(.+),匹配任意符号，会尽可能多的匹配

**需要将贪婪模式修改为懒惰模式**

|  |
| --- |
| //使用量词约束筛选的字符数量  $str = '<img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894714.png"><img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894715.png">';    //需求：将上面字符中的图片地址筛选出来  $reg = '/<img[^>]\*src="(.+?)">/us'; //（//u斜线外面的u表示采用万能点模式）  //使用规则去字符串中筛选  *preg\_match*($reg, $str,$match);  **echo** '<pre>';  *var\_dump*($match); |



### preg\_match\_all()

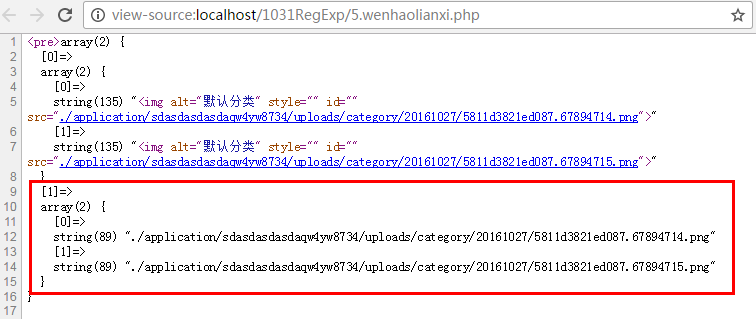
（4）如何筛选所有的图片呢？

preg\_match\_all()，和preg\_match的区别在于，

preg\_match只匹配一次，所以当preg\_match()匹配到 14.png”>的时候，就停止

preg\_match\_all() 会一直匹配到字符串的结尾

|  |
| --- |
| //使用量词约束筛选的字符数量  $str = '<img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894714.png"><img alt="默认分类" style="" id="" src="./application/sdasdasdasdaqw4yw8734/uploads/category/20161027/5811d3821ed087.67894715.png">';    //需求：将上面字符中的图片地址筛选出来  $reg = '/<img[^>]\*src="(.+?)">/us';  //使用规则去字符串中筛选  *preg\_match\_all*($reg, $str,$match);  **echo** '<pre>';  *var\_dump*($match); |



练习：

这里有一些字符串

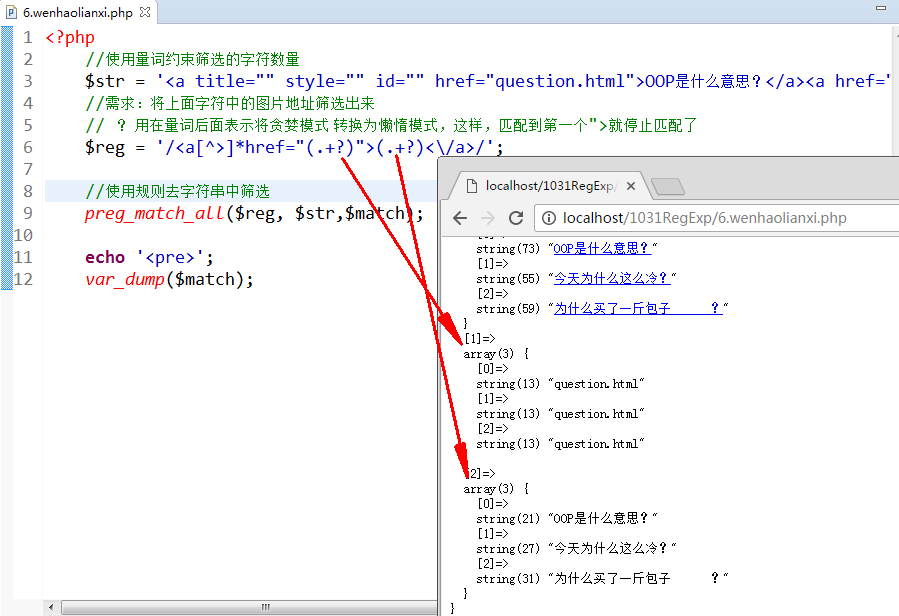
|  |
| --- |
| $str = '<a href="question.html">OOP是什么意思？</a><a href="question.html">今天为什么这么冷？</a><a href="question.html">为什么买了一斤包子 ？</a>'; |

大家将该字符串中超链接的文本内容部分、链接的href属性值筛选出来

分组的使用：

我们通过()将需要匹配的内容，使用（）括起来，返回结果的时候，第一个小括号匹配的内容就会保存到数组的下标为1的元素里面，第二个小括号匹配的内容就会保存到数组的下标为2的元素里面，依次类推

**懒惰模式会遍历循环 贪婪模式就一次不循环**

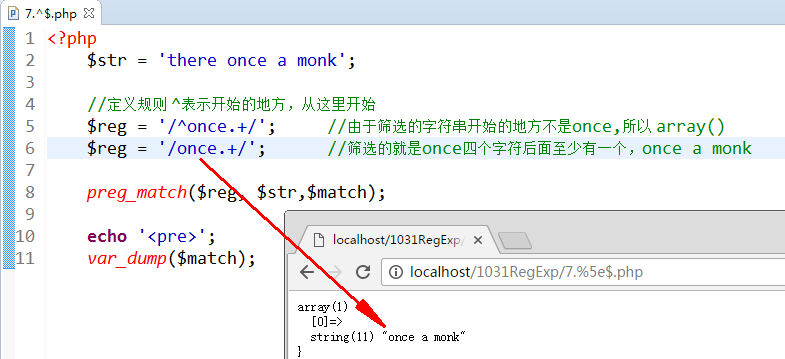


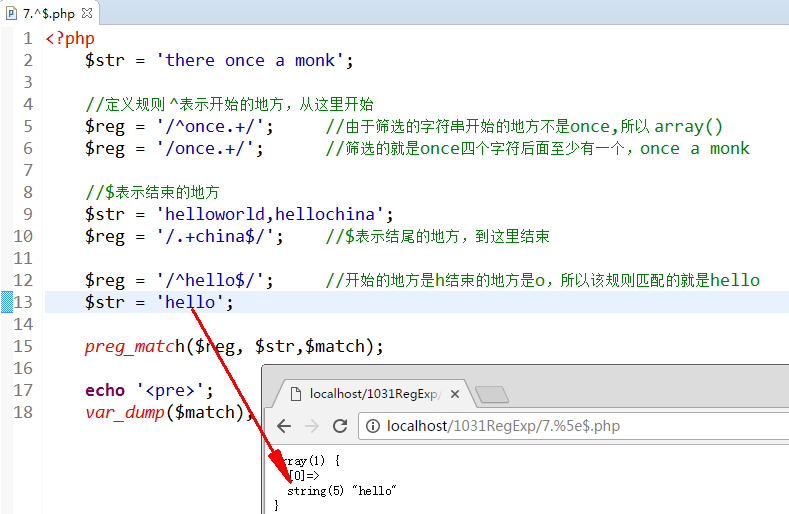
### ^$

^ $特殊符号的使用

^ 在[]里面表示非，除了...之外的字符

^ 在[]外面，表示从这里开始，开始的地方，$表示在这里结束，结束的地方





^ $ 通常匹配字符的时候，匹配的更精确

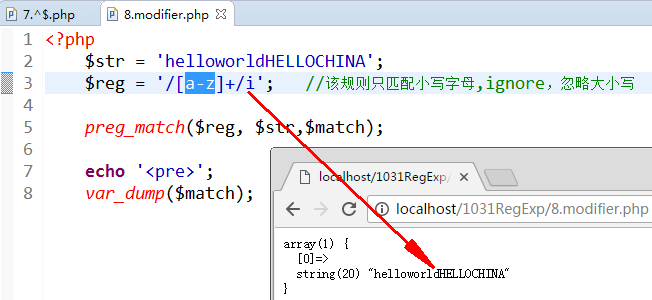
## 模式修订符

模式修订符是对前面的正则表达式的一个修饰

例如：

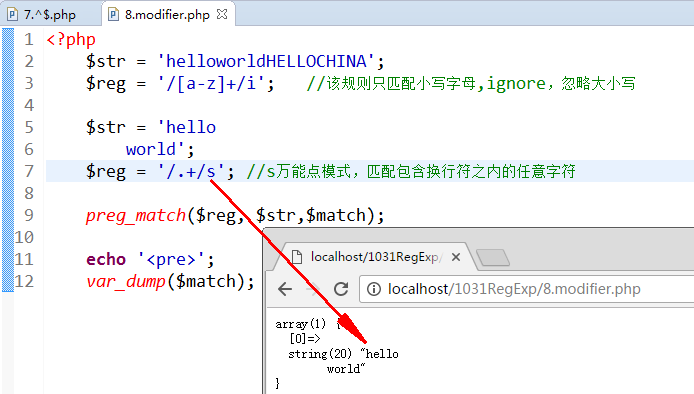
### //i

修饰前面的正则表达式，将其修改为忽略大小写 ignore



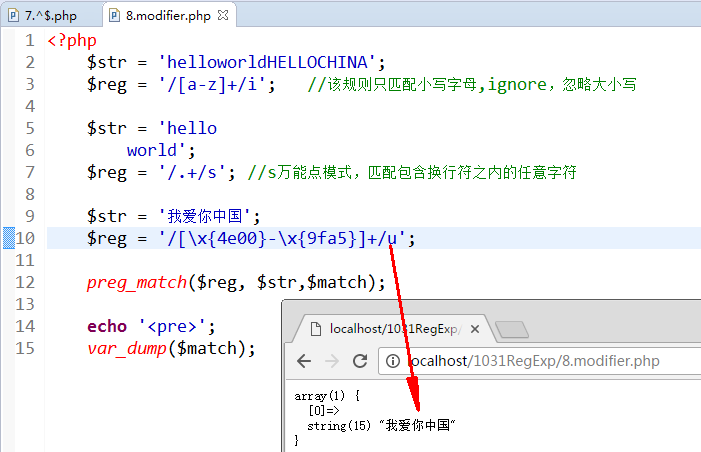
### //s

万能点模式，将前面的正则表达式修饰为万能点模式，匹配任意字符，包括换行



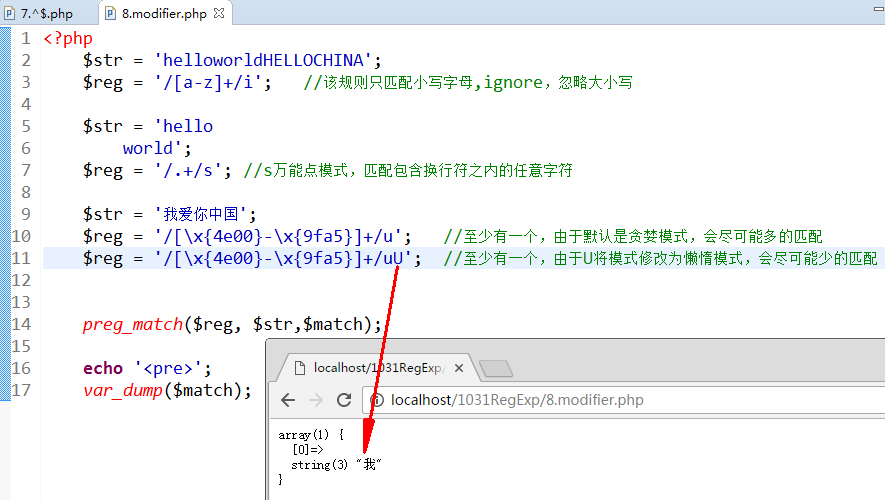
### //u

**，采用utf8编码处理**



### //U

大写的U，表示将前面的正则表达式修饰为 采用懒惰模式

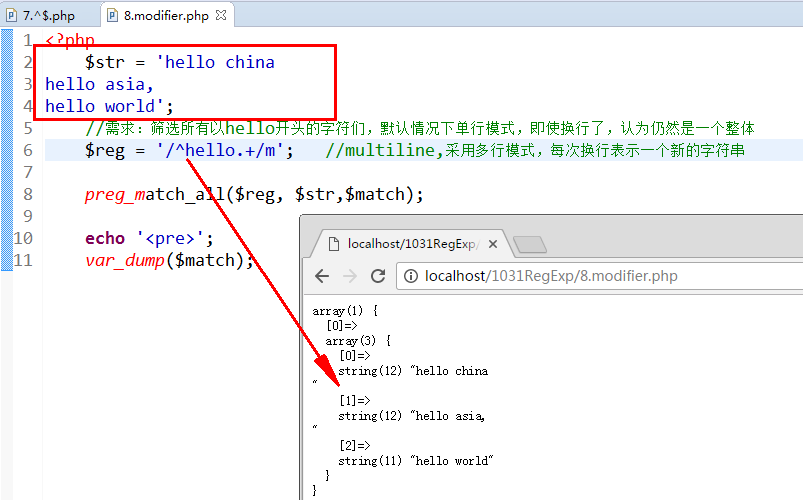


### //m

multime line 将正则表达式修饰为多行模式

默认情况，即使一个字符出现换行，也认为这是一个字符串

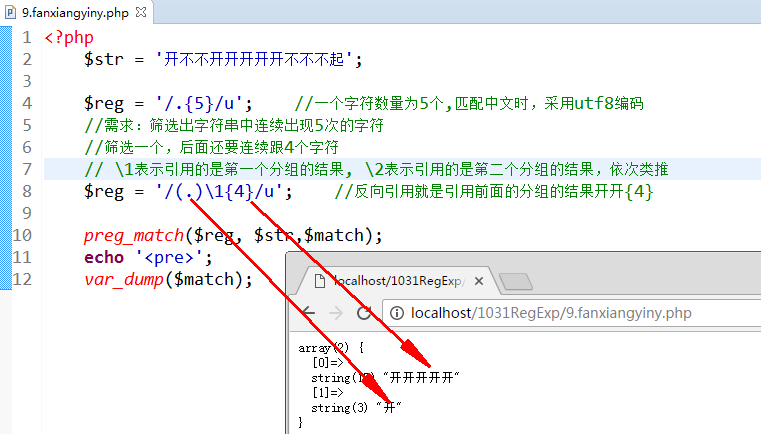
使用 m 表示多行模式，每次换行表示一个新的字符串了



## 反向引用

### \1

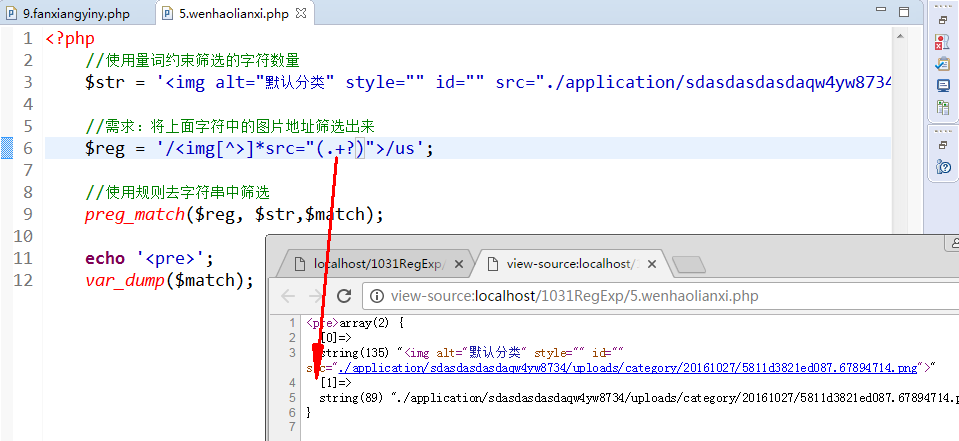
就是引用的前面分组的结果，\1引用（使用）前面第一个分组的结果，\2引用的是前面第二个分组的结果....以此类推



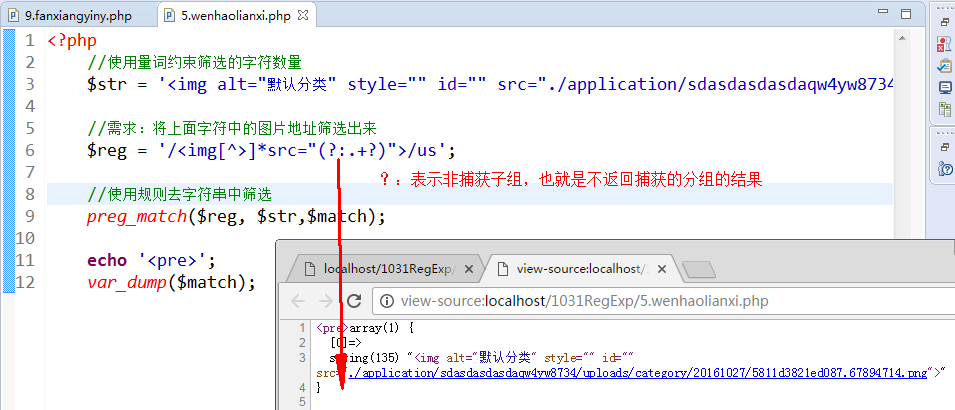
## 非捕获子组

不返回某个分组的结果

使用非捕获子组之前



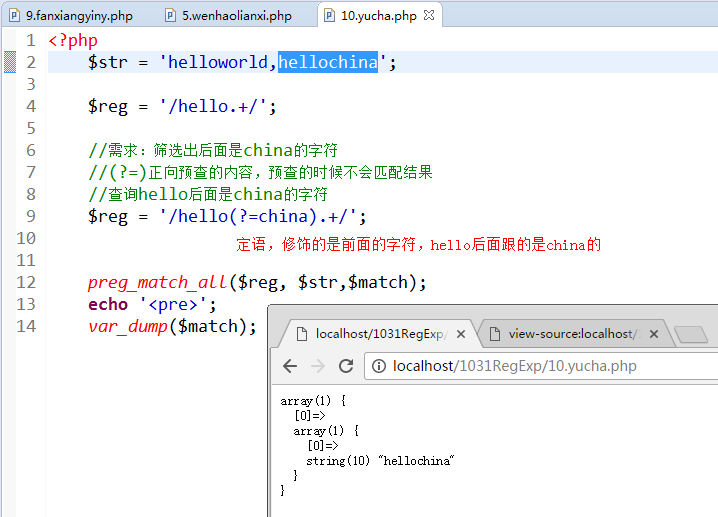
**使用非捕获子组之后的结果：**



## 正向预查

什么是正向：从左向右

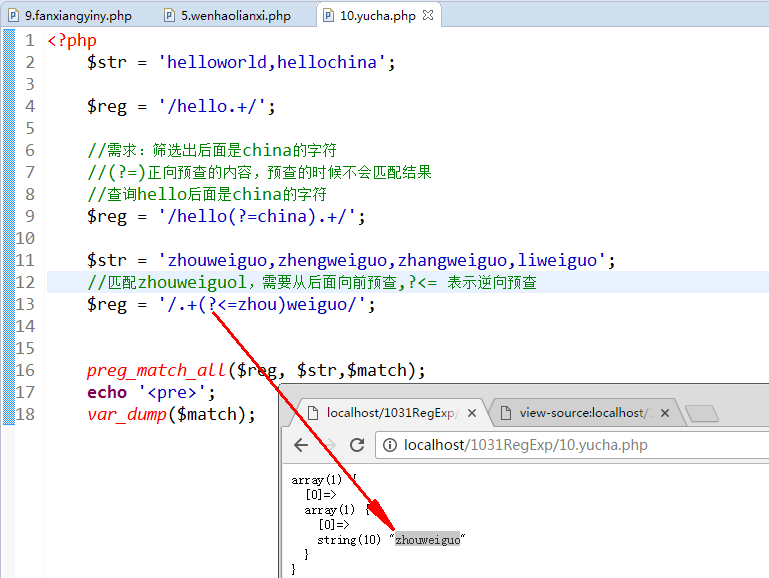
预查：提前查一下



## 逆向预查

逆向：从右向左

预查，提前踩点，提前查询一下



## 例子

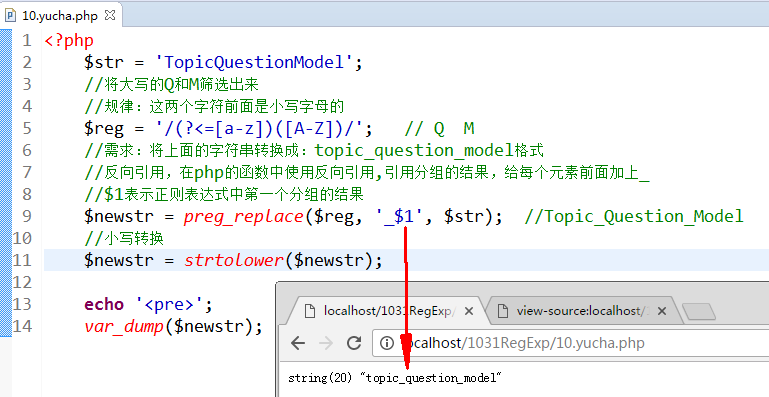
练习一把：

我们有一个字符串：

$str = 'TopicQuestionModel';

//将大写的Q和M筛选出来

大家可以参考上面正向预查、逆向预查实现



## 正则表达式相关的函数

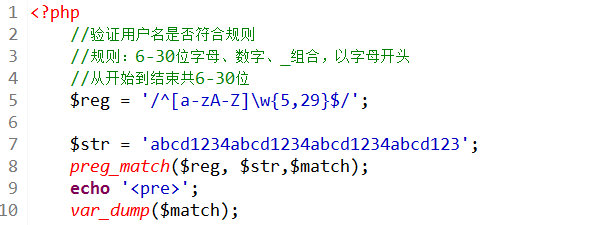
|  |
| --- |
| //参数1：正则表达式规则（筛子）  //参数2：筛选的数据，海量数据  //参数3：将筛选的内容保存到第三个参数  //preg\_match($reg,$str,$match)  $str = 'hello world';  $reg = '/.\*/';  //preg\_match($reg,$str,$match);  //var\_dump($match);    //preg\_match\_all() 和 preg\_match()的 区别：  //preg\_match()只要有一个符合规则，将结果筛选出来之后就停止  //preg\_match\_all() 会一直匹配到字符串结果，将所有符合规则的都筛选出来  $str = 'TopicQuestionModel';  $reg = '/(?<=[a-z])([A-Z])/';  *preg\_match\_all*($reg, $str,$match);      //preg\_replace()  //参数1：正则表达式规则  //参数2：替换的结果  //参数3：在哪个字符串中进行查找并替换  $newstr = *preg\_replace*($reg, '\_$1', $str); //Topic\_Question\_Model    //preg\_split()  //split()字符串拆分的意思  $str = 'i love china;i love php!i love laohan';  $reg = '/[;!]/';  $res = *preg\_split*($reg, $str);  **echo** '<pre>';  *var\_dump*($res); |

## 正则表达式案例

### 验证用户名

是否符合规则

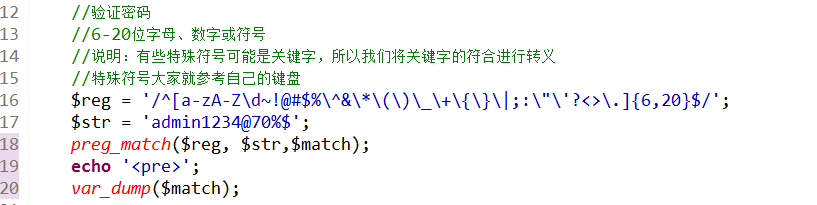
规则：6-30位字母、数字、\_组合，以字母开头



### 验证用户输入的密码

是否符合规则

规则：6-20位字母、数字或符号



### 验证用户输入的邮箱

是否符合规则

