实验 5-补充案例

案例5-1 体验正则表达式

一. 案例描述

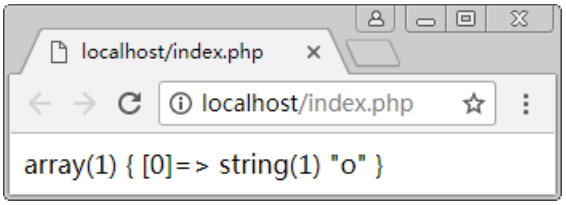
1. **考核知识点**
   正则表达式语法
2. **练习目标**
   - 掌握正则表达式在 PHP 中的使用
3. **需求分析**
   在 PHP 文件中定义一个正则表达式，用于匹配一个指定字符。
4. **设计思路（实现原理）**
   1） 创建一个 PHP 文件，定义一个变量$pattern 保存正则表达式。
   2） 定义一个变量$subject，该变量为要匹配的字符串。
   3） 使用 preg_match()函数进行匹配，将匹配结果以数组的形式返回并显示。

二. 案例实现

1. 创建 PHP 文件，定义正则表达式并调用 preg_match()函数。

```
<?php
$pattern = '/o/';
$subject = 'hello world';
preg_match($pattern, $subject, $matches);
var_dump($matches);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

1. 定义一个正则表达式一般使用“/”表示正则的起始位置和结束位置。
2. 在 PHP 中编写正则表达式需要使用引号，即定义成一个字符串。
3. 正则表达式“/o/”用于匹配小写字母 o。
4. preg_match()函数在成功匹配到 1 个元素后就会停止匹配，即使后面仍然有符合匹配条件的字符也

不会再进行匹配。

案例5-2 字符范围

一. 案例描述

1. **考核知识点**

   正则表达式字符范围的使用

2. **练习目标**

   - 掌握正则表达式定义字符范围的语法规则
   - 掌握正则表达式字符范围的使用

3. **需求分析**

在正则表达式中，有时需要匹配一个范围内的字符，例如匹配 26 个小写字母，如果使用“abcde…..xyz”的方式太过繁琐，因此可以利用字符范围来实现。

4. **设计思路（实现原理）**

   1） 创建一个 PHP 文件，定义一个变量$pattern 保存正则表达式。

   2） 该正则表达式匹配“f-p”这个范围内的字符。

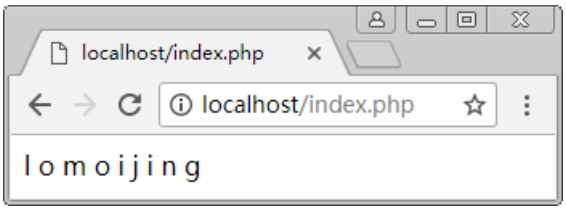
   3） 使用 preg_match_all()函数进行匹配，然后输出匹配结果。

二. 案例实现

1. 利用正则表达式匹配目标内容。

```
<?php
$pattern = '/[f-p]/';
$subject = 'Welcome to Beijing';
preg_match_all($pattern, $subject, $matches);
echo implode(' ', $matches[0]);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

1. 定义字符范围时，使用中括号“[”和“]”将字符范围包裹，字符范围取最前一个和最后一个，中间以连字符“-”连接，例如 [a-z]。
2. 如果只是想要匹配某几个不连续的字符中的某一个，可以写成[ascd]这种形式。
3. 字符范围遵循 ASCII 中的顺序，不能颠倒。
4. 字符范围可以连写，如希望匹配 a-f、j-r 这两个范围内的字符可以写成[a-fj-r]。

5. 可以匹配数字范围，例如[a-z0-9]表示匹配全部小写字母和十进制数字中的任何一个。

案例5-3 定位符“^”

一. 案例描述

1. **考核知识点**

   定位符“^”的使用

2. **练习目标**

   ➢ 掌握定位符“^”在正则表达式中的使用

3. **需求分析**

   在程序开发中，经常需要筛选出字符串起始位置是某些特定字符的字符串，此时定义正则表达式就需要用到“^”符号，例如有一组英文名，需要找到其中以 n 开头的英文名。

4. **设计思路（实现原理）**

   1）创建一个 PHP 文件，定义数组$name，保存以下几个英文名：neala、ben、jhonson、nancy、nitya、thomas、jenna。

   2）定义一个正则表达式用于筛选首字母为 n 的英文名，使用 /^n/ 筛选。

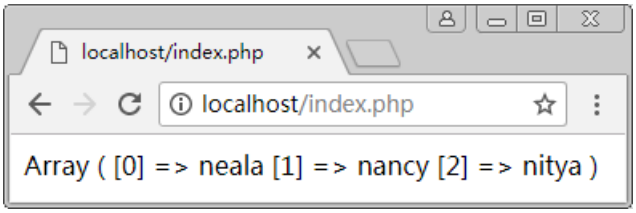
   3）遍历$name 数组，将符合筛选条件的英文名保存到一个新数组中，然后打印出来。

二. 案例实现

1. 创建 PHP 文件，定义正则表达式筛选符合条件的元素。

```
<?php
$pattern = '/^n/';
$name = ['neala', 'ben', 'johnson', 'nancy', 'nitya', 'thomas', 'jenna'];
$matches = [];
for ($i = 0, $len = count($name); $i < $len; ++$i) {
    if (preg_match($pattern, $name[$i])) {
        $matches[] = $name[$i];
    }
}
print_r($matches);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

在正则表达式中，使用“^”符号可以匹配字符串的开始位置。

案例5-4 定位符“$”

一. 案例描述

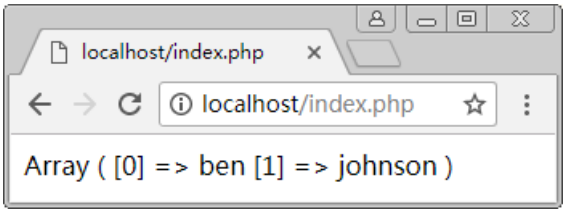
1. **考核知识点**
   定位符“$”的使用
2. **练习目标**
   - 掌握正则表达式中定位符“$”的使用
3. **需求分析**
   定位符“$”的作用与“^”正好相反，用于筛选出字符串末尾是某些特定字符的字符串。
4. **设计思路（实现原理）**
   1）创建一个 PHP 文件，定义数组$name，其中包含有以下几个英文名：neala、ben、jhonson、nancy、nitya、thomas、jenna。
   2）定义一个正则表达式，用于筛选末尾字母为 n 的英文名。
   3）遍历$name 数组，将符合筛选条件的英文名存放到一个新数组中并显示，分析两种正则表达式筛选结果有何不同。

二. 案例实现

1. 创建 PHP 文件，定义正则表达式筛选符合条件的数值。

```
<?php
$pattern = '/n$/';
$name = ['neala', 'ben', 'johnson', 'nancy', 'nitya', 'thomas', 'jenna'];
$matches = [];
for ($i = 0, $len = count($name); $i < $len; ++$i) {
    if (preg_match($pattern, $name[$i])) {
        $matches[] = $name[$i];
    }
}
print_r($matches);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

1. “^”符号表示的是匹配字符串的开始，而“$”符号表示匹配字符串的结束。
2. 如果要匹配一个空行，使用 /^$/ 这个正则表达式即可。
3. 对于“^”符号有一点需要注意，当该符号在“[]”中出现时，将不用于表示开始位置，此时的“^”符号是对字符范围取反操作，例如 /[^0-8]/ 可以匹配数字 9，不能匹配数字 0～8。

案例5-5 特殊字符“*”、“+”、“?”

一. 案例描述

1. **考核知识点**

   特殊字符“*”、“+”、“?”的使用。

2. **练习目标**

   - 掌握特殊字符“*”、“+”、“?”的使用。

3. **需求分析**

   在表单验证中，经常需要使用正则表达式对数据进行验证，例如验证用户注册的用户名是否合法，一般会规定用户名由哪些字符组成，此时在正则表达式中就会用到特殊字符“*”、“+”、“?”。

4. **设计思路（实现原理）**

   1）在 HTML 文件中创建一个表单，该表单将用户注册的用户名提交给 check.php。

   2）在 check.php 中接收数据，利用正则表达式来判断用户名是否符合规定。

   3）使用 preg_match()函数调用正则进行判断，如果符合规定则输出“用户名合法”，否则输出“用户名不合法”。

二. 案例实现

1. 在 form.html 文件中创建表单，该表单将提交给 check.php。

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>用户注册表单</title>
  </head>
  <body>
    <form method="post" action="check.php">
      用户名：<input type="text" name="name"><br>
      <font color="red">用户名必须以字母开头，可以包含数字</font><br>
      <input type="submit" value="注册"><input type="reset" value="清除">
    </form>
  </body>
</html>
```

2. 编写 check.php，用于接收表单后对用户名进行验证。具体代码如下。

```
<?php
$pattern = '/^[a-zA-Z]+[a-zA-Z0-9]*$/';
$subject = $_POST['name'];
if (!preg_match($pattern, $subject)) {
    echo '用户名不合法';
} else {
    echo '用户名合法';
}
```

3. 案例结果：

如果传入的用户名不是以字母开头则会输出“用户名不合法”。

如果传入的用户名是以字母开头，之后字母或数字出现 0 次或多次都会输出“用户名合法”。

三. 案例总结

1. 特殊符号“*”的作用是匹配其前面的单元 0 次或多次。
2. 特殊符号“+”的作用是匹配其前面的单元 1 次或多次。
3. 特殊符号“?”的作用是匹配其前面的单元 0 次或 1 次。
4. 一般在进行这种字符规则验证的时候，都要使用“^”符号和“$”符号以保证全部目标内容都匹配，而不是只匹配一部分。

案例5-6 特殊字符“.”

一. 案例描述

1. **考核知识点**

特殊字符“.”

2. **练习目标**

- 掌握特殊字符“.”的使用

3. **需求分析**

在利用正则表达式匹配字符时，有时候需要匹配任何字符，这时就需要使用特殊字符“.”。

4. **设计思路（实现原理）**

1） 创建 PHP 文件，定义正则表达式匹配任意字符串。

2） 准备多个不同的字符串用于测试，字符串中可以出现任何字符。

3） 使用 preg_match()函数调用正则表达式，然后显示匹配结果。

二. 案例实现

1. 根据“.”符号的特殊作用，结合“*”符号可以实现匹配任意字符串的功能。

```
<?php
$pattern = '/.*/';
$subject = ['*&^%dicn134ich?/-d+', "[你好]\nbeijing", 'sic(d89c+--__)kecia~'];
```

```
$matches = [];
for ($i = 0, $len = count($subject);$i < $len; ++$i) {
    if (preg_match($pattern, $subject[$i])) {
        $matches[] = $subject[$i];
    }
}
print_r($matches);
```

2. 通过浏览器访问，查看 HTML 源代码，运行结果如下图所示。

三. 案例总结

利用正则表达式 /.*/ 可以对任何字符串进行匹配。

案例5-7 转义字符“\”

一. 案例描述

1. **考核知识点**

   转义字符“\”的使用

2. **练习目标**

   - 掌握转义字符“\”的使用

3. **需求分析**

   在正则表达式中，有时候需要匹配一些特殊字符，但这些特殊字符在正则表达式语法中又有其特定的作用，这时就需要使用转义字符“\”对特殊字符进行转义。

4. **设计思路（实现原理）**

   1） 定义一个变量，变量的值为“tom like $”。

   2） 定义正则表达式，实现完全匹配上面的变量。

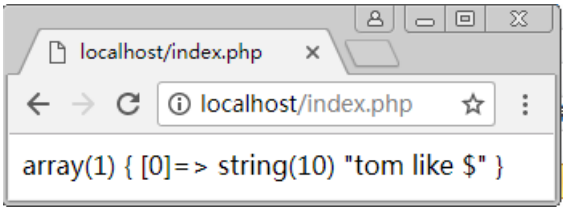
   3） 使用 preg_match()函数执行正则表达式匹配，然后将匹配结果显示。

二. 案例实现

1. 在 PHP 文件中声明变量并赋值，在定义正则用以匹配该变量。

```
<?php
$pattern = '/tom like \$/';
$subject = 'tom like $';
preg_match($pattern, $subject, $matches);
var_dump($matches);
```

2. 通过浏览器访问，查看 HTML 源代码，运行结果如下图所示。



三. 案例总结

1. 转义符“\”可以将特殊字符的特定含义忽略，只保留字符本身的原意。
2. 如果需要匹配“\”，使用“\\”即可，其他特殊字符同样在其前面添加“\”符号。
3. 需要注意的是，在 PHP 中通过字符串书写转义字符时，PHP 代码本身也会转义一次。

案例5-8 选择字符“|”

一. 案例描述

1. **考核知识点**

   选择字符“|”的使用

2. **练习目标**

   - 掌握选择字符“|”的使用

3. **需求分析**

在验证用户数据的时候，经常需要验证某一位置的几个字符是否符合规范。例如，验证邮箱的域名，若域名只允许仅有的几种组合，这时可以使用选择字符“|”来进行匹配。

4. **设计思路（实现原理）**

   1）在 HTML 文件中创建一个表单，该表单将用户登录使用的邮箱地址以 POST 方式提交给 check.php。

   2）在 check.php 中接收数据，通过正则表达式来判断邮箱是否符合规定。

   3）使用 preg_match()函数进行匹配，如果符合筛选条件则输出“邮箱合法”，否则输出“邮箱不合法”。

二. 案例实现

1. 在 form.html 中创建表单，该表单将以 POST 方式提交给 check.php。

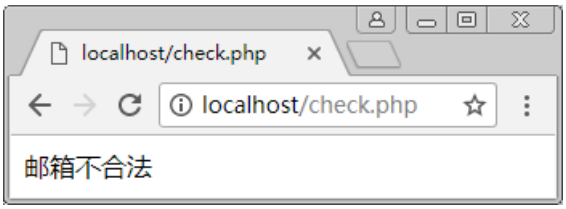
```
<!doctype html>
<html>
  <head>
```

```
    <meta charset="utf-8">
    <title>邮箱验证</title>
  </head>
  <body>
    <form method="post" action="check.php">
      登录邮箱：<input type="text" name="email"><br>
      <input type="submit" value="登录">
      <input type="reset" value="清除">
    </form>
  </body>
</html>
```

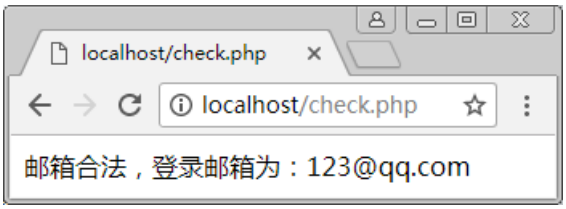
2. 编写 check.php，接收表单进行验证。具体代码如下。

```
<?php
$subject = $_POST['email'];
$pattern = '/[a-zA-Z0-9]+@(qq.com|hotmail.com|live.cn|163.com|126.com|sina.com)$/';
preg_match($pattern, $subject, $matches);
if (!$matches) {
    echo '邮箱不合法';
} else {
    echo '邮箱合法，登录邮箱为：' . $matches[0];
}
```

3. 如果输入一个不存在与正则表达式中的邮箱地址，例如 123@test.com：



4. 如果输入存在于正则表达式中的邮箱地址，运行结果如下。



三. 案例总结

1. 此处邮箱验证仅是举例说明“|”的使用，真实邮箱的验证规则要更复杂。
2. 括号字符“()”在此处的作用是改变限定符的作用范围。

案例5-9 preg_match()函数

一. 案例描述

1. 考核知识点

   preg_match()函数的使用

2. 练习目标

   ➢ 掌握 preg_match()函数的使用

3. 需求分析

在程序开发中，经常需要根据正则表达式的模式对指定的字符串进行搜索匹配。这时，可以使用preg_match()函数。

4. 设计思路（实现原理）

   1） 声明一个变量，为该变量赋值一个任意长度的字符串。

   2） 定义一个正则表达式，用于匹配变量。

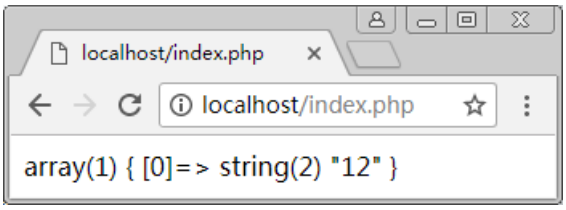
   3） 使用 preg_match()函数调用正则表达式，显示匹配结果。

二. 案例实现

1. 在 PHP 文件中定义变量，通过正则表达式进行匹配。

```
<?php
$subject = 'na12sicmea92xisax3';
$pattern = '/[0-9]{2}/';
preg_match($pattern, $subject, $matches);
var_dump($matches);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

preg_match()函数在成功匹配到 1 个元素后就会停止匹配，即使后面仍然有符合匹配条件的字符。

案例5-10 preg_match_all()函数

一. 案例描述

1. 考核知识点

preg_match_all()函数的使用

2. 练习目标

- 掌握 preg_match_all()函数的使用

3. 需求分析

对于正则表达式“/[0-9]{2}/”，字符串“na12sicmea92xisax3”有两处符合匹配条件，分别是 12 和 92，但 preg_match()函数在成功匹配到第 1 个，也就是 12 后就停止匹配。如果需要匹配整个目标中全部符合条件的内容，则需要使用 preg_match_all()函数。

4. 设计思路（实现原理）

1） 利用 preg_match_all()函数执行正则表达式匹配。

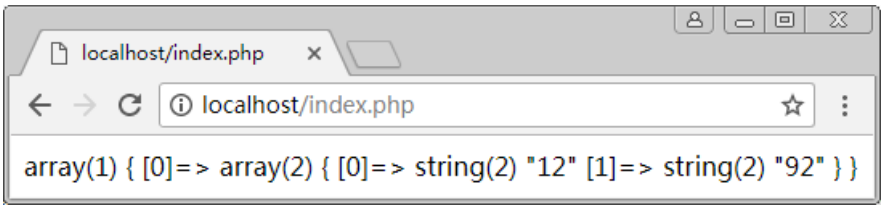
2） 对比 preg_match_all()函数和 preg_match()函数的区别。

二. 案例实现

1. 编写 index.php，利用正则表达式进行匹配。

```
<?php
$subject = 'na12sicmea92xisax3';
$pattern = '/[0-9]{2}/';
preg_match_all($pattern, $subject, $matches);
var_dump($matches);
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

preg_match_all()函数会一直匹配到最后才停止，获取到的所有匹配结果以二维数组的形式返回。

案例5-11　preg_grep()函数

一. 案例描述

1. 考核知识点

preg_grep()函数的使用

2. 练习目标

- 掌握 preg_grep()函数的使用

3. 需求分析

在程序开发中，经常需要使用正则表达式对数组中的元素进行匹配，此时可以使用 preg_grep()函数。

4. 设计思路（实现原理）

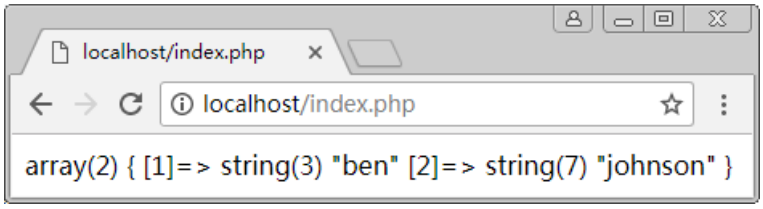
1） 利用 preg_grep()函数对数组进行正则表达式匹配。
2） 输出匹配结果。

二. 案例实现

1. 编写 index.php，具体代码如下。

```
<?php
$pattern = '/n$/';
$name = ['neala', 'ben', 'johnson', 'nancy', 'nitya', 'thomas', 'jenna'];
var_dump(preg_grep($pattern, $name));
```

2. 通过浏览器访问，运行结果如下图所示。



三. 案例总结

preg_grep()函数可以对数组中的元素进行匹配。

案例5-12　preg_split()函数

一. 案例描述

1. 考核知识点

preg_split()函数的使用

2. 练习目标

- 掌握 preg_split()函数的使用

3. 需求分析

在开发中经常需要对中文字符串进行截取，而中文字符属于多字节字符，使用 substr()函数会出现乱码。PHP 提供了一种处理多字节字符的函数 mb_substr()可以解决这个问题，但需要 PHP 加载了 mbstring 扩展。其实，利用正则表达式可以也实现多字节字符的处理，下面将演示 preg_split()函数的使用。

4. 设计思路（实现原理）

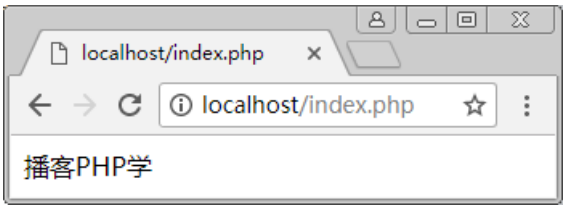
1） 声明一个 substr_utf8()函数。
2） 该函数有 3 个参数，第 1 个为要处理的字符串，第 2 个为截取开始位置，第 3 个为要截取的长度。
3） 将目标字符串使用 preg_split()函数利用正则表达式分割成数组中的元素。
4） 使用 array_slice()函数根据传入的截取开始位和截取长度，从该数组中取出一段。
5） 使用 implode()函数将截取出来的数组重新转化成字符串，然后返回。

二. 案例实现

1. 声明函数 substr_utf8()，该函数将截取无乱码的中文字符串。

```
<?php
/**
 * 截取 UTF-8 字符集的字符串，支持中文
 * @param string $str 要处理的字符串
 * @param int $start 从哪个位置开始截取
 * @param int $length  要截取字符串的个数
 * @return string 截取后的字符串
 */
function substr_utf8($str, $start, $length)
{
  return implode('', array_slice(preg_split('//u', $str, -1, PREG_SPLIT_NO_EMPTY),
$start, $length));
}
$str = '传智播客 PHP 学院';
echo substr_utf8($str, 2, 6);
```

2. 案例运行结果如下图所示：



三. 案例总结

1. 由于数组的数字键名从 0 开始计算，所以开始位置 2 的中文字符为“播”。
2. preg_split()函数有 2 个必选参数，第 1 个为用于搜索的模式，也就是正则表达式，第 2 个为要匹配的目标，字符串类型。第 3 个参数为可选参数，如果指定，则将显示分割得到的子串最多只能有指定的个数。最后 1 个参数也是可选参数，具体细节请参考手册。
3. array_slice()函数是获取数组中的一段，有 3 个参数，第 1 个为目标，数组类型，第 2 个为获取的数组起始下标，第 3 个参数为获取数组元素的个数。