جامعة الانبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم أنظمة شبكات الحاسوب

# برمجة صفحات الويب PHP

المرحلة الثالثة

الفصل الدراسي الاول

مدرس المادة

م.د. سميه عبدالله حمد

## Lectures:

١. lec١: PHP Fundamentals (اساسيات برمجة صفحات الانترنت)
٢. lec٢: PHP Data Types (أنواع البيانات)
٣. lec٣: PHP Arrays (المصفوفات)
٤. lec٤: PHP Logic Control Structures (عبارات السيطرة)
٥. lec٥: PHP Loops (عبارات التكرار)
٦. lec٦: PHP Strings PHP String Functions (السلاسل ودوالها)
٧. lec٧: PHP Function (الدوال)
٨. lec٨: PHP Date() & Time Function (دوال الوقت والتاريخ)
٩. lec٩: Regular Expressions (التعابير المنتظمة)
١٠. lec١٠: PHP File Handling & Functions (دوال وتحميل الملفات)
١١. lec١١: PHP Session & PHP Cookies ( الجلسات وملفات تعريف الارتباط)
١٢. lec١٢: How to Send Email using PHP mail (كيفية ارسال الايميل)

# 5- PHP preg_match(): Regular Expressions (Regex)

## What is Regular expression in PHP?

**PHP Regular Expression** also known as regex are powerful pattern matching algorithm that can be performed in a single expression. Regular expressions use arithmetic operators such as (+,-,^) to create complex expressions. They can help you accomplish tasks such as validating email addresses, IP address etc.

## Why use regular expressions

- PHP Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time.
- When validating user input such as email address, domain names, telephone numbers, IP addresses,
- Highlighting keywords in search results
- When creating a custom HTML template. Regex in PHP can be used to identify the template tags and replace them with actual data.

## Built-in Regular expression Functions in PHP

PHP has built in functions that allow us to work with regular functions. The commonly used regular expression functions in PHP.

- preg_match() in PHP – this function is used to perform pattern matching in PHP on a string. It returns true if a match is found and false if a match is not found.
- preg_split() in PHP – this function is used to perform a pattern match on a string and then split the results into a numeric array
- preg_replace() in PHP – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as PHP preg_match(), PHP preg_split() or PHP preg_replace().

```
<?php
function_name('/pattern/',subject);
?>
```

HERE,

- “function_name(…)” is either PHP preg_match(), PHP preg_split() or PHP preg_replace().
- “/…/” The forward slashes denote the beginning and end of our PHP regex tester function
- “‘/pattern/'” is the pattern that we need to matched
- “subject” is the text string to be matched against

Let’s now look at practical examples that implement the above PHP regex functions.

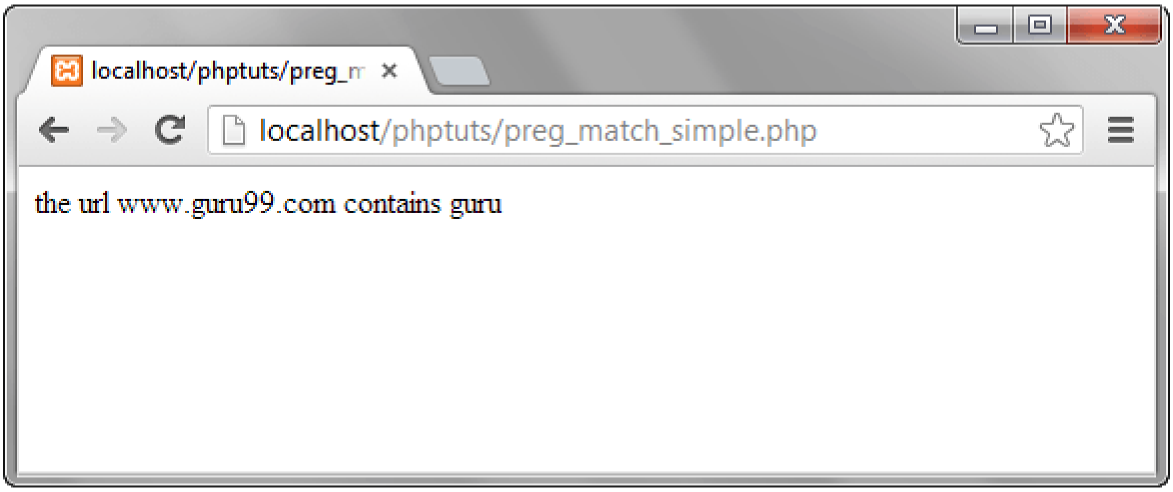
## Preg_match() in PHP

The first example uses the preg_match() in PHP function to perform a simple pattern match for the word guru in a given URL.

The code below shows the implementation for preg_match() tester function for the above example.

```
<?php
$my_url = "www.guru99.com";
if (preg_match("/guru/", $my_url))
{
        echo "the url $my_url contains guru";
}
else
{
        echo "the url $my_url does not contain guru";
}
?>
```

Browse to the URL **http://localhost/phptuts/preg_match_simple.php**



the url www.guru99.com contains guru

Let’s examine the part of the code responsible for our output “*preg_match(‘/guru/’, $my_url)*” HERE,

- “preg_match(…)” is the PHP regex function
- ““/guru/”” is the regular expression pattern to be matched
- “$my_url” is the variable containing the text to be matched against.

## PHP Preg_split()

Let’s now look at another example that uses the preg_split() in PHP function.

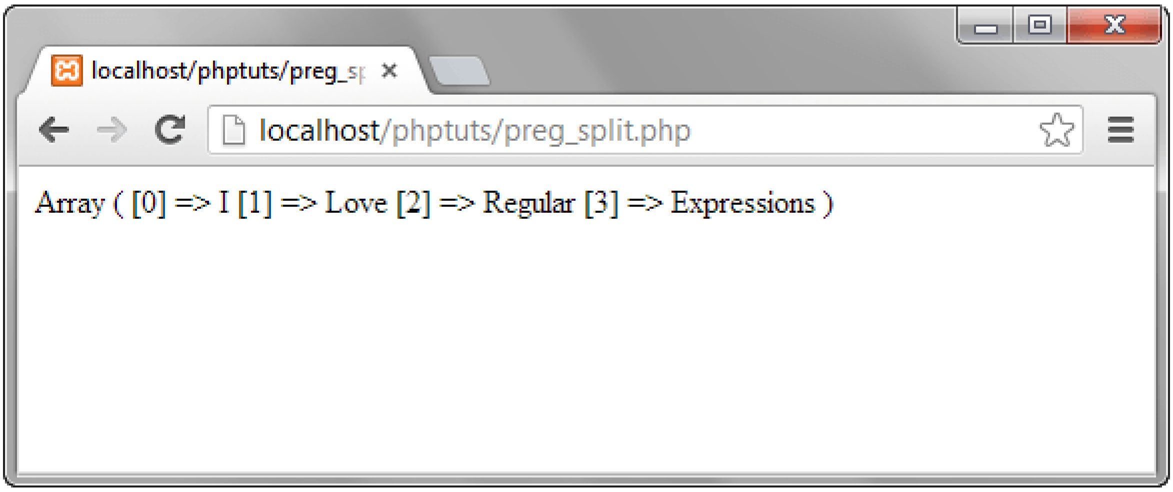
We will take a string phrase and explode it into an array; the pattern to be matched is a single space.

The text string to be used in this example is “I Love Regular Expressions”.

The code below illustrates the implementation of the above example.

```
<?php
$my_text="I Love Regular Expressions";
$my_array  = preg_split("/ /", $my_text);
print_r($my_array );
?>
```

Browse to the URL **http://localhost/phptuts/preg_split.php**



```
Array ( [0] => I [1] => Love [2] => Regular [3] => Expressions )
```

## PHP Preg_replace()

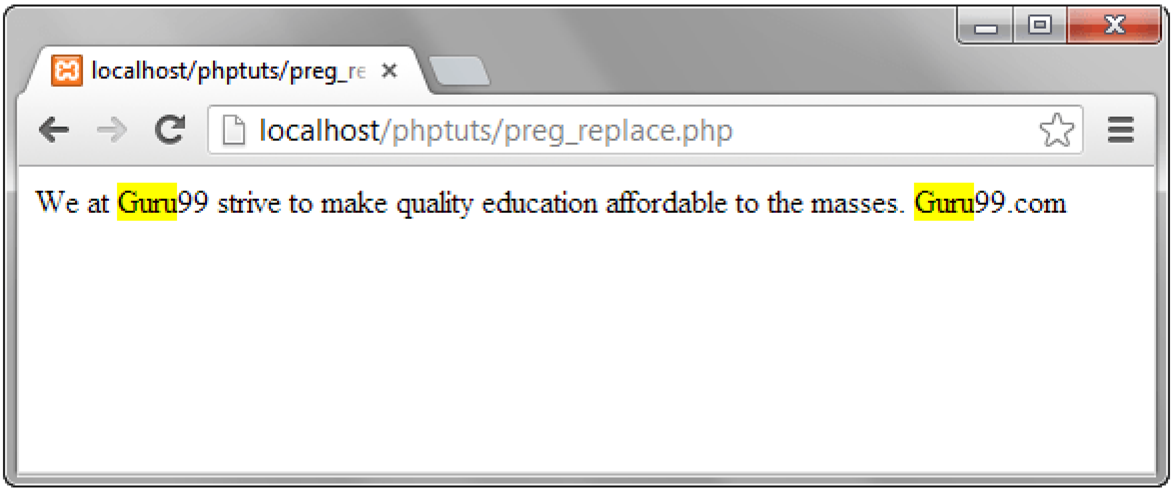
Let’s now look at the preg_replace() in PHP function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word guru in a string.

It replaces the word guru with the word guru surrounded by css code that highlights the background colour.

```
<?php
$text = "We at Guru99 strive to make quality education
affordable to the masses. Guru99.com";
$text = preg_replace("/Guru/", '<span
style="background:yellow">Guru</span>', $text);
echo $text;
?>
```

Assuming you have saved the file preg_replace.php, browser to the URL **http://localhost/phptuts/preg_replace.php**



We at Guru99 strive to make quality education affordable to the masses. Guru99.com

## Regular Expression Metacharacters

The above examples used very basic patterns; metacharacters simply allow us to perform more complex pattern matches such as test the validity of an email address. Let’s now look at the commonly used metacharacters.

| Metacharacter | Description | Example |
|---|---|---|
| . | Matches any single character except a new line | /./ matches anything that has a single character |
| ^ | Matches the beginning of or string / excludes characters | /^PH/ matches any string that starts with PH |
| $ | Matches pattern at the end of the string | /com$/ matches guru99.com,yahoo.com Etc. |
| * | Matches any zero (0) or more characters | /com*/ matches computer, communication etc. |
| + | Requires preceding character(s) appear at least once | /yah+oo/ matches yahoo |
| \ | Used to escape meta characters | /yahoo+\.com/ treats the dot as a literal value |
| […] | Character class | /[abc]/ matches abc |
| a-z | Matches lower case letters | /a-z/ matches cool, happy etc. |

| Metacharacter | Description | Example |
|---|---|---|
| A-Z | Matches upper case letters | /A-Z/ matches WHAT, HOW, WHY etc. |
| 0-9 | Matches any number between 0 and 9 | /0-4/ matches 0,1,2,3,4 |

The above list only gives the most commonly used metacharacters in regular expressions.

Let’s now look at a fairly complex example that checks the validity of an email address.

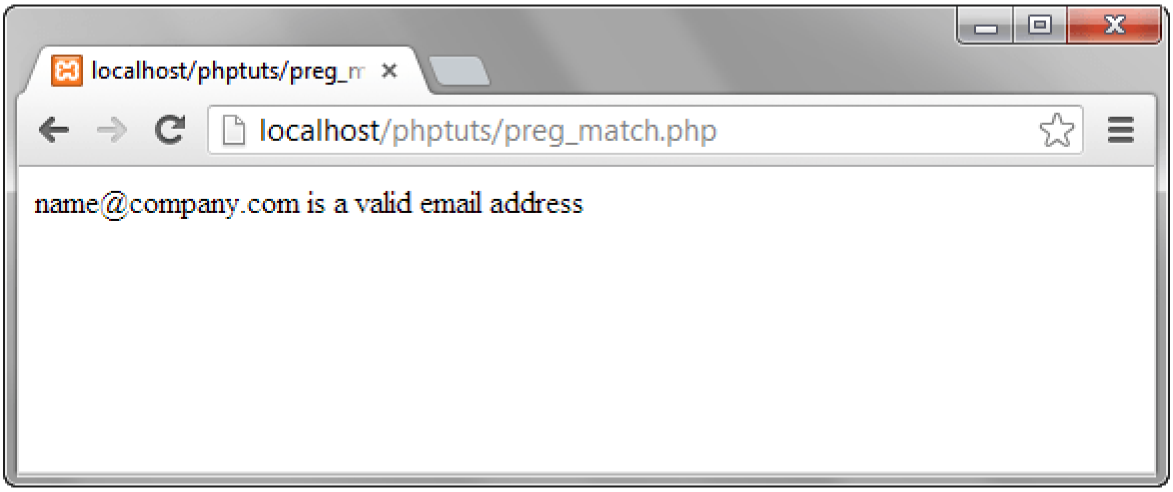
```
<?php
$my_email = "name@company.com";
if (preg_match("/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-
Z.]{2,5}$/", $my_email)) {
echo "$my_email is a valid email address";
}
else
{
  echo "$my_email is NOT a valid email address";
}
?>
```

Explaining the pattern “[/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/]” HERE,

- “‘/…/’” starts and ends the regular expression
- “^[a-zA-Z0-9._-]” matches any lower or upper case letters, numbers between 0 and 9 and dots, underscores or dashes.
- “+@[a-zA-Z0-9-]” matches the @ symbol followed by lower or upper case letters, numbers between 0 and 9 or dashes.

- "+\.[a-zA-Z.]{2,5}$/" escapes the dot using the backslash then matches any lower or upper case letters with a character length between 2 and 5 at the end of the string.

Browse to the URL **http://localhost/phptuts/preg_match.php**



name@company.com is a valid email address

As you can see from the above example breakdown, metacharacters are very powerful when it comes to matching patterns.

## Summary

- A Regular Expression or Regex in PHP is a pattern match algorithm
- Regular expressions are very useful when performing validation checks, creating HTML template systems that recognize tags etc.
- PHP has built in functions namely PHP preg_match(), PHP preg_split() and PHP preg_replace() that support regular expressions.
- Metacharacters allow us to create complex patterns