Advanced PHP

Prof. Jayakumar Sadhasivam,
School of Information Technology and Engineering,
Vellore Institute of Technology, Vellore.

PHP
Regular Expressions

PHP - Regular Expressions

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

Regular Expression Functions

Function	Description
preg_match()	Returns **1** if the pattern was found in the string and **0** if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string
preg_grep()	Returns an array consisting only of elements from the input array which matched the
preg_split()	The preg_split() function breaks a string into an array using matches of a regular expression as separators.

preg_match()

```
Syntax: preg_match(pattern,
   input,matches,flags,offset)
```

Pattern	**Required**. Contains a regular expression indicating what to search for
Input	**Required**. The string in which the search will be performed
Matches	**Optional**. The variable used in this parameter will be populated with an array containing all of the matches that were found
Flags	**Optional**. A set of options that change how the matches array is structured
Offset	**Optional.** Defaults to 0. Indicates how far into the string to begin searching.

preg_match()

```
<?php
  $str = 'Visit jayakumars.in website.
     Its jayakumar personal website.';
  $pattern = '/jayakumar/';
  echo preg_match( $pattern, $str );
?>
Output: 1
```

preg_match_all()

```
Syntax: preg_match_all
   (pattern, input, matches,
   flags, offset)
```

Pattern	**Required**. Contains a regular expression indicating what to search for
Input	**Required**. The string in which the search will be performed
Matches	**Optional**. The variable used in this parameter will be populated with an array containing all of the matches that were found
Flags	**Optional**. A set of options that change how the matches array is structured
Offset	**Optional.** Defaults to 0. Indicates how far into the string to begin searching.

preg_match_all()

```
<?php
  $str = 'Visit jayakumars.in website.
     Its jayakumar personal website.';
  $pattern = '/jayakumar/';
  echo preg_match_all( $pattern,
     $str );
?>
```

```
Output: 2
```

preg_replace()

```
Syntax: preg_replace(patterns,
   replacement, input, limit, count)
```

Pattern	**Required**. Contains a regular expression or array of regular expressions
Replacements	**Required**. A replacement string or an array of replacement string
Input	**Required**. The string or array of strings in which replacements are being performed
Limit	**Optional**. Defaults to -1, meaning unlimited. Sets a limit to how many replacements can be done in each string.
Count	**Optional.** After the function has executed, this variable will contain a number indicating how many replacements were performed

preg_replace()

```
<?php
  $str = "Visit hello.in!";
  $pattern = "/hello/";
  echo preg_replace($pattern, “jayakumars",
       $str);
?>
```

```
Output: Visit jayakumars.in!
```

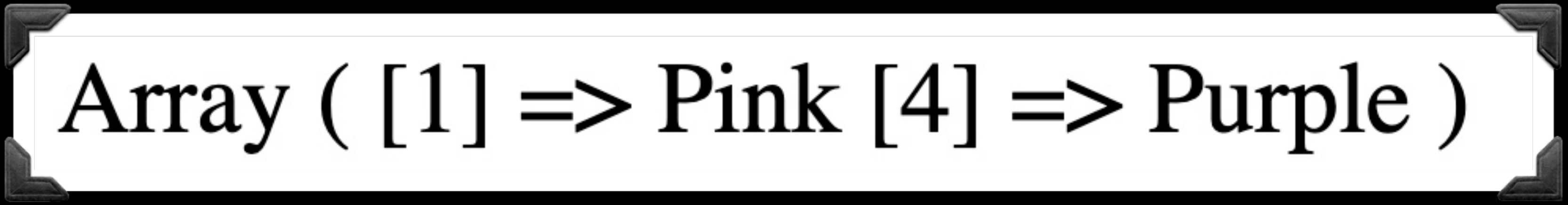
preg_grep()

Syntax: `preg_grep(pattern, input,flags)`

Pattern	**Required**. Contains a regular expression indicating what to search for
Input	**Required**. An array of strings
Flags	**Optional**. There is only one flag for this function. only items that do not match the pattern.

preg_grep()

```php
<?php
//method 1
$input = ['Red','Pink','Green','Blue','Purple'];
//method 2
$input = array( 'Red', 'Pink', 'Green', 'Blue',
   'Purple' );
//find the word starting with letter P
$result = preg_grep( '/^p/i', $input );
print_r( $result );
?>
```



```
Array ( [1] => Pink [4] => Purple )
```

Regular Expressions

- The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.

^	Starts With
$	Ends With
?	Zero or One
*	Zero or More
+	One or More
[ ]	Range Symbol
{ }	Limit Symbol
( )	Group Symbol

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	Description
i	Makes the match case insensitive
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns
g	Globally finds all matches
cg	Allows a search to continue even after a global match fails
o	Evaluates the expression only once

Regular Expression Patterns

Brackets are used to find a range of characters	
Modifier	**Description**
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9
[a-zA-Z]	Find one character from a to z and A-Z

Metacharacters

Metacharacters are characters with a special meaning	
Metacharacter	**Description**
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
. (dot)	Matches any single character except newline
^ (caret)	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$

Metacharacters

Metacharacters are characters with a special meaning

Metacharacter	Description
\d	Find a digit (0-9)
\D	Matches any non-digit character
\s	Matches any whitespace character (space, tab, newline or carriage return character). Same as [\t\n\r] (lowercase S)
\S	Matches any non-whitespace character. Same as [^ \t\n\r] (Uppercase S)
\w	Matches any word character (definned as a to z, A to Z,0 to 9, and the underscore.
\W	Matches any non-word character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b (whole word or exact word).

Quantifiers

Quantifiers define quantities	
Quantifiers	**Description**
n+	Matches one or more occurrences of the letter n
n*	Matches zero or more occurrences of the letter n
n?	Matches zero or one occurrences of the letter n
n{2}	Matches exactly two occurrences of the letter p.
n{2,5}	Matches at least two occurrences of the letter n, but not more than five occurrences of the letter n.
n{2,}	Matches two or more occurrences of the letter n
n{,3}	Matches at most three occurrences of the letter n

Example

- Let us say "a" is a given string

a? = {},{a}

a* = {},{a},{aa},{aaa} ...

a+ = {a},{aa},{aaa} ...

^ab = {ab}

[1-3] = {1,2,3}

{1,3} = Minimum 1 , Maximum 3

Username

```
$uname = ‘Jayakumars';
if ( preg_match( "/^[a-zA-Z0-9_]
   {3,50}$/", $uname ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
Output: Matched
```

Password

```
$pwd = 'Jaya#123$kumar';
if ( preg_match( "/^[a-zA-Z0-9*&^%
   $#@!-]{8,20}$/", $pwd ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
Output: Matched
```

Email

```
$email = 'jayakumars@vit.ac.in';
if ( preg_match( "/^[a-z0-9._-]+@[a-z]
   +.[a-z.]{2,5}$/", $email ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
```

```
Output: Matched
```

Date

```
$date = ‘01-01-2021';
if ( preg_match( "/(^[0-9]
   {1,2})-[0-9]{1,2}-([0-9]
   {4}$)/", $date ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
Output: Matched
```

Mobile Number

```
$mobile = '9876543210';
$mobile = '09876543210';
if ( preg_match( "/(^[+0-9]{1,3})?([0-9]
   {10,11}$)/", $mobile ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
Output: Matched
```

URL

```
<?php
  $url = 'www.jayakumars.in';
  if ( preg_match( "/(^([www.]?)[0-9a-
   z_.]+.[a-z.]{1,5}$)/", $url ) ) {
    echo 'Matched';
  } else {
    echo 'Not matched';
  }
  //Output: Matched
?>
```

Pincode

```
$pincode = 632014;
if ( preg_match( "/^[0-9]{6}$/",
   $pincode ) ) {
     echo 'Matched';
} else {
     echo 'Not matched';
}
Output: Matched
```

E-Mail

Emailing with PHP

- If you are in a *NIX environment, you will most likely have sendmail installed on your server.
- If you are using a hosting service, check with your service provider to make sure sendmail or some equivalent is being used.
- Once you have your mail server squared away, you'll need to modify your php.ini

★ **SMTP** - Set this to the ip address or DNS name of your SMTP server. default value : localhost

★ **sendmail_from** -The From address used by default by the PHP mail() command

★ **smtp_port** - Set this to the port PHP uses to connect to the SMTP server. default value 25

mail() function

- PHP makes use of mail() function to send an email.
- This function requires three mandatory arguments that specify the recipient's email address, the subject of the the message and the actual message additionally there are other two optional parameters.

```
mail( to, subject, message, headers, parameters )
```

mail() function

Parameter	Description
to	**Required**. Specifies the receiver / receivers of the email
subject	**Required.** Specifies the subject of the email. This parameter cannot contain any newline characters
message	**Required.** Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. (LF - Line Feed)
headers	**Optional**. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). (CRLF - Carriage Return Line Feed)
parameters	**Optional.** Specifies an additional parameter to the send mail program

Simple Mail program

```php
<?php
$to="user123@abc.com";
$sub="Test Mail";
$msg = "Sample Test Mail";
$mailsent = mail($to,$sub,$msg);
if($mailsent){
  echo "the mail was sent to $to
           successfully";  }
else {
  echo "the mail couldn’t send. Pl verify your
           mail settings / connection";  }
?>
```

Collecting Data & Sending Email

- Can collect the data through HTML forms and can send an email.
- Should create HTML program to collect information like to-address, subject, message from user and PHP program to send mail using these information.

HTML Mail Program

```
<html>
<form action="email.php" method="POST">
  EmailId:
  <input type="text" name="email" size=40>
  <br>Subject:
  <input type="text" name="subject" size=40>
  <br>Message:<br>
  <textarea cols=40 rows=10 name=“message">
  </textarea>
  <input type="submit" value="Send">
</form>
</html>
```

Mail() function using HTML

```
<?php
ini_set("sendmail_from", "user@abc.com");
ini_set("SMTP","mail.abc.com");
echo "<h1>Welcome to emailing with PHP</h1>";
echo "<br>";
$to= $_REQUEST["to"];
$subject= $_REQUEST["subject"];
$message= $_REQUEST["MESSAGE"];
$from = "user@abc.com";
$headers = "From:" . $from;
$send=mail($to,$subject,$message,$headers);
```

Mail() function using HTML

```
if($send){
 echo "congrats! The following Mail has been
             Sent<br><br>";
 echo "<b>To:</b> $to<br>";
 echo "<b>From:</b> $from<br>";
 echo "<b>Subject:</b> $subject<br>";
 echo "<b>Message:</b><br>";
 echo $message;
}
else{
 echo "error in sending mail....";
}
?>
```

Mail to Many Address

```
<?php
$to="user1@mail.com, abc1@mail.com,
  abc2@mail.com";
$sub="Test Mail";
$msg = "Sample Test Mail";
$mailsent = mail($to,$sub,$msg);
if($mailsent){
  echo "the mail was sent to $to successfully";  }
else {
  echo "the mail couldn’t send. Pl verify your
    mail settings / connection";  }
?>
```

Email with CC & BCC

The cc & bcc addresses to be included in header parameter as follows:

$header="cc: user1@abc.com, user2@aa.com"

Or

$header="bcc: user1@abc.com, user2@aa.com"

If u want to add both together, then

$header ="cc: user1@abc.com, user2@aa.com\r\n"

$header.="bcc: user1@abc.com, user2@aa.com"

Here \r\n – is the boundary between cc & bcc. It is must to use \r\n to the header, if you want to add more parameters.

Email with CC

```
<? php
$to="user123@abc.com";
$sub="Test mail";
$message1 = "This is a sample test mail";
$header = "cc:aa@aa.com, ab@aa.com";
$mailsent = mail($to,$sub,$message1,$header);
if ($mailsent){
    echo "the mail was sent to $to and
       $headersuccessfully”;}
else{
    echo "the mail couldn’t send. Pl verify your
       mail settings / connection”; }
?>
```

Email with CC and BCC

```
<? php
$to="user123@abc.com";
$sub="Test mail";
$message1 = "This is a sample test mail";
$header = "cc:aa@aa.com, ab@aa.com\r\n";
$header •="bcc: ac@aa.com";
$mailsent = mail($to,$sub,$message1,$header);
if ($mailsent){
    echo "the mail was sent to $to and
        $headersuccessfully”;}
else{
    echo "the mail couldn’t send. Pl verify your mail
        settings / connection”; }
?>
```

Email – HTML Messages

- In order to send messages as a plain text and an HTML, will use headers with additional information as follows:
  - ✓ `$headers  = "MIME Version 1.0 \r\n";`
  - ✓ `$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";`
  - ✓ `$headers .= "Content-Transfer-Encoding: 7bit\r\n";`
- Tell the e-mail client that data is coming in multiple parts—in this instance, plain text and HTML. This tells the e-mail client to look for additional "Content-type" information in the message, which includes boundary information. The boundary is what separates the multiple parts of the message.

Email – HTML messages

```
<?php
$to="user123@abc.com";
$sub="Test mail";
$message1 = "<h1>MIME mail</h1>This<br> is <br>a <br>sample
    <br>test <br>mail";
$headers = "MIME-Version: 1.0\r\n";
$headers •= "Content-type: text/html;
    charset=iso-8859-1\r\n";
$headers •= "Content-Transfer-Encoding: 7bit\r\n";
$mailsent = mail($to,$sub,$message1,$header);
If ($mailsent)
    echo "the mail was sent to $to successfully";
Else
    echo "mail was not sent";
?>
```

Sending attachments with email

- To send a email with an attachment, should do the following steps:
  - ✓ Step 1: Open the attach file and transfer its content to a php variable.
  - ✓ Step 2: Assign that file content to the message parameter of the mail function with some attributes and boundary which are needed for attachment.
  - ✓ Step 3: Now add the message which has to be delivered to the receiver with attributes to specify it is a message.
- Note: Here, every information has to be separated with a boundary value.

Sending attachments with email

- To send an email with mixed content requires to set Content-type header to multipart/mixed. Then text and attachment sections can be specified within boundaries.
- A boundary is started with two hyphens followed by a unique number which can not appear in the message part of the email.
- A PHP function md5() is used to create a 32 digit hexadecimal number to create unique number.
- A final boundary denoting the email's final section must also end with two hyphens.

Email with an attachment

1. **Open the file** which has to be attached with the mail in rb mode and transfer the content into a php variable.

```
$fileatt = “mysql_php.pdf"; // Path to the file
$file = fopen($fileatt,'rb');
$data = fread($file,filesize($fileatt));
fclose($file);
```

2. Set the attributes for attachment

   2.1. Now add the content-type to the message as the type of attached file.

```
$email_message = “--{$mime_boundary}\n".
   "Content-Type: {$fileatt_type};\n”;
```

Email with an attachment

2. Set the attributes for attachment

2.2. Then can rename the attach file as follows:

```
$email_message•= "name=\"{$fileatt_name}\"\n";
```

(here same name used, you can change the name here, if you want.)

2.3. Add “content-disposition: attachment” – which indicates the mail is coming with an attachment.

```
$email_message •= "Content-Disposition:
        attachment;\n";
```

2.4. Add the transfer encoding type.

```
$email_message •= "Content-Transfer-Encoding:
        base64\n\n";
```

Email with an attachment

2. Set the attributes for attachment

2.5. Now it is a time to add the content of the attachment file.

```
    $email_message .= $data . "\n\n";
```

2.6. Add the boundary with the message as a ended one.

```
  $email_message .= "--{$mime_boundary}--\n";
```

3. Now can add message which has to send as a plain or HTML:

```
$email_message .= "your  file has been sent as
      an <h1>attachment </h1>to the <font
      color = red> user specified by you</
      font>";
```

Email with an attachment

4. Now add attributes for message

```
$email_message •= "This is a multi-part message
    in MIME format.\n\n" ."--{$mime_boundary}\n" .
"Content-Type:text/html;
    charset=\"iso-8859-1\"\n" .
"Content-Transfer-Encoding: 7bit\n\n" .
```

After done all the above use mail() as follows:

```
$headers = "\nMIME-Version: 1.0\n" ."Content-
    Type: multipart/mixed;\n" ."
    boundary=\"{$mime_boundary}\"";
$sent=mail($to, $sub, $email_message,$headers);
```

Email with an attachment

Detail explanation about previous steps:

1. Collect the message and store it in a variable.
2. Add the boundary at the end of that message.
3. Enclose the content type of that message and its transfer encoding method.
4. Now add boundary to mark it as end of message.
5. Open the file which has to be attached with the mail in r+ mode and transfer the content into a php variable.
6. Now add the content-type to the message as the type of attached file.

Email with an attachment

7. Then set the name as the file name which has to be sent an attachment.
8. Add “content-disposition: attachment” – which indicates the mail is coming with an attachment.
9. Can add the rename option of the attached file.
10. Add the transfer encoding type.
11. Now it is a time to add the content of the attachment file
12. Add the boundary with the message as a ended one.

Chunk_split

- The chunk_split() function splits a string into a series of smaller parts.
- Syntax
  - ✓ chunk_split(string,length,end)

- ParameterDescription
  - ✓ string Required. Specifies the string to split
  - ✓ Length Optional. A number that defines the length of the chunks. Default is 76
  - ✓ end Optional. A string that defines what to place at the end of each chunk. Default is \r\n
- Note: This function does not alter the original string.

Sample program

```
<?php
// IMAGE ATTACHMENT
$fileatt = "test1.jpg"; // Path to the file
$fileatt_type = "image/jpg"; // File Type
$fileatt_name = "test1.jpg"; // Filename that will
    be used for the file as the attachment

//TEXT ATTACHMENT
$fileatt = "MYSQL-PHP.pdf"; // Path to the file
$fileatt_type = "application/pdf"; // File Type
$fileatt_name = "PHPebook.pdf"; // Filename that
    will be used for the file as the attachment
```

Sample program

```
// READ THE FILE
$file = fopen($fileatt,'r+');
$data = fread($file,filesize($fileatt));
fclose($file);

// ENCODE THE FILE WHICH HAS TO ATTACH
$data = chunk_split(base64_encode($data));
```

Sample program

```
// ASSIGNING ADDRESSES, SUBJECT & MESSAGE
$email_from = “user@bac.com";
// Who the email is from
//$email_subject = "image file attached"; // The
   Subject of the email
$email_subject= "PHP book attached as a pdf file";
$email_message = "Thanks for visiting mysite.com!
   Here is your free file.<br>";
$email_message •= "Thanks for visiting.<br>”;
 // Message that the email has in it
$email_to = “user123@bac.com”;
  // Who the email is to
```

Sample program

```
//CREATE HEADER FOR MULTIPART MESSAGE
$headers = "From: ".$email_from;
$semi_rand = md5(time());
$mime_boundary = "==Multipart_Boundary_x{$semi_rand}x";

$headers .= "\nMIME-Version: 1.0\n" .
"Content-Type: multipart/mixed;\n" .
" boundary=\"{$mime_boundary}\"";

// DEFINING MESSAGE TYPE
$email_message .= "This is a multi-part message in MIME format.
    \n\n" .
"--{$mime_boundary}\n" .
"Content-Type:text/html; charset=\"iso-8859-1\"\n" .
"Content-Transfer-Encoding: 7bit\n\n" .
$email_message .= "\n\n";
```

Sample program

```
//ATTACHING THE FILE
$email_message .= "--{$mime_boundary}\n" .
"Content-Type: {$fileatt_type};\n" .
" name=\"{$fileatt_name}\"\n" .
"Content-Disposition: attachment;\n" .
" filename=\"{$fileatt_name}\"\n" .
"Content-Transfer-Encoding: base64\n\n" .
$data .= "\n\n" .
"--{$mime_boundary}--\n";

//SEND MAIL
$ok = mail($email_to, $email_subject,
   $email_message, $headers);
```

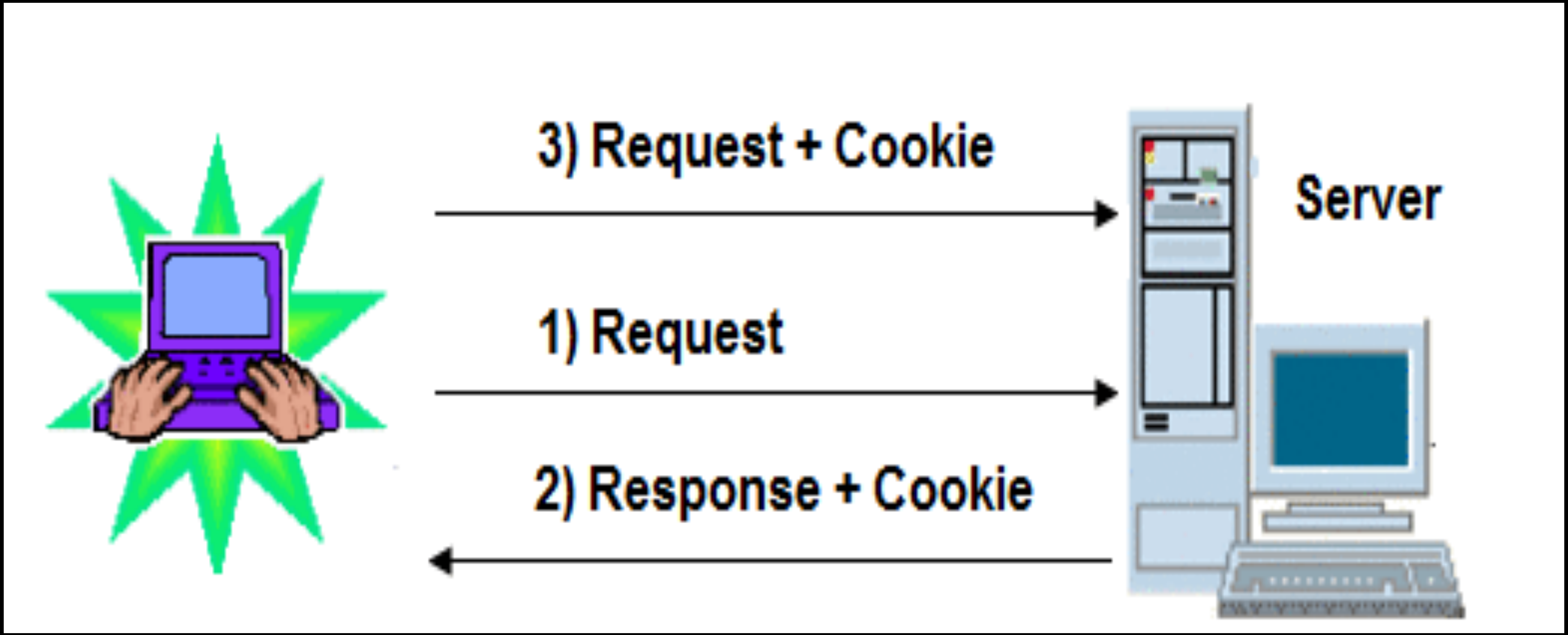
Sample program

```
// check the mail was sent or not
if($ok) {
echo "Your file has been sent<br> to the email
   address you specified.<br> Make sure to
   check your junk mail!<br>";
}
else
{
die("Sorry but the email could not be sent.
   Please go back and try again!");
}
?>
```

COOKIE

HTTP Cookies

- Cookies are arbitrary pieces of data, usually chosen and first sent by the web server, and stored on the client computer by the web browser.
- The browser then sends them back to the server with every request, introducing states (memory of previous events) into otherwise stateless HTTP transactions.
- Without cookies, each retrieval of a web page or component of a web page would be an isolated event, largely unrelated to all other page views made by the user on the website.


3) Request + Cookie
Server
1) Request
2) Response + Cookie

HTTP Cookies

- The cookie specifications require that browsers meet the following requirements in order to support cookies:
  - ✓ Can support cookies as large as 4,096 bytes (4 KB) in size.
  - ✓ Can support at least 50 cookies per domain (i.e. per website).
  - ✓ Can support at least 3,000 cookies in total.

What is a Cookie?

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With PHP, you can both create and retrieve cookie values.

Setting a Cookie in PHP

- The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set. The basic syntax of this function can be given with

```
setcookie(name, value, expire, path,
       domain, secure);
```

Setting a Cookie in PHP

Parameter	Description
name	The name of the cookie.
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

Creating a Cookie in PHP

```
<?php
// Setting a cookie
setcookie("username", "Jayakumar",
   time()+30*24*60*60);
?>
```

setcookie() function to create a cookie named username and assign the value value Jayakumar to it. It also specify that the cookie will expire after 30 days (30 days * 24 hours * 60 min * 60 sec)

Accessing Cookies Values

- The PHP $_COOKIE superglobal variable is used to retrieve a cookie value.
- It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name.
- The individual cookie value can be accessed using standard array notation, for example to display the username cookie use the following code.

```php
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

Check Cookie is Set or Not

- It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:

```
<?php
// Verifying whether a cookie is set or not
if(isset($_COOKIE["username"])){
    echo "Hi " . $_COOKIE["username"];
} else{
    echo "Welcome Guest!”;  }
?>
```

- You can use the print_r() function like print_r($_COOKIE); to see the structure of this $_COOKIE associative array, like you with other arrays.

Removing Cookies

- You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past, as shown in the example below:

```php
<?php
// Deleting a cookie
setcookie("username","", time()-3600);
?>
```

SESSION

What is a Session?

- A session is a way to store information (in variables) to be used across multiple pages.
- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.

What is a Session?

- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

Why and when to use Sessions?

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

Session

- The session IDs are randomly generated by the PHP engine .
- The session data is stored on the server therefore it doesn’t have to be sent with every browser request.
- The session_start() function needs to be called at the beginning of the page, before any output is generated by the script in the browser.

Creating a Session

- To begin a new session, simply call the PHP **session_start()** function. It will create a new session and generate a unique session ID for the user.
- Session variables are set with the PHP global variable: $_SESSION.

```
<?php
session_start();
?>
```

- The session_start() function first checks to see if a session already exists by looking for the presence of a session ID.
- If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

Storing and Accessing Session Data

- You can store all your session data as key-value pairs in the $_SESSION[] superglobal array.
- The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

```
<?php
session_start();
// Storing session data
$_SESSION["firstname"] = "Jayakumar";
$_SESSION["lastname"] = "Sadhasivam";
?>
```

Storing and Accessing Session Data

- To access the session data from any other page on the same web domain — simply recreate the session by calling session_start() and then pass the corresponding key to the $_SESSION associative array.

```
<?php
session_start();

// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] .' '
    . $_SESSION["lastname"];
?>
```

O/P : Hi, Jayakumar Sadhasivam

Destroying a Session

- If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array.

```
<?php
// Starting session
session_start();

// Removing session data
if(isset($_SESSION["lastname"]))
{
    unset($_SESSION["lastname"]);
}
?>
```

Destroying a Session

- However, to destroy a session completely, simply call the session_destroy() function. This function does not need any argument and a single call destroys all the session data.

```
<?php
// Starting session
session_start();

// Destroying session
session_destroy();
?>
```

Session Page Count

```php
<?php
session_start();
if(isset($_SESSION[‘page_count']))
{
     $_SESSION['page_count'] += 1;   }
else {
    $_SESSION['page_count'] = 1;
}
 echo 'You are visitor number ' .
   $_SESSION['page_count'];
?>
```

Thank You !!!

😎