4.2 PHP COOKIES and REGULAR EXPRESSIONS IN PHP

***Cookies are small data stored on the client computer and they are kept of use tracking purpose*.**

PHP COOKIES

PHP transparently supports HTTP cookies. There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

➢ **Setting Cookies with PHP:**

PHP provided **setcookie()** function to set a cookie. This function requires six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

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

- **Name:** This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value:** This sets the value of the named variable and is the content that the user wants to store.
- **Expiry:** This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the browser is closed.
- **Path**: This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain**: This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security:** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Setting a cookie

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
  setcookie("age", "36", time()+3600, "/", "", 0); ?>
<html><head> <title>Setting Cookies with PHP</title> </head>
<body>
<?php echo "Set Cookies"?> </body></html>
```

- **Accessing Cookies with PHP**

  PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables.isset() function is used to check if a cookie is set or not.

Accessing cookies

```
<html><head> <title>Accessing Cookies with PHP</title> </head>
<body> <?php
echo $_COOKIE["name"]. "<br />";
/* is equivalent to */
echo$HTTP_COOKIE_VARS["name"]. "<br/>";
echo $_COOKIE["age"] . "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"] . "<br />";
?> </body></html>
```

- **Deleting Cookie**

  To delete a cookie users should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on. It is safest to set the cookie with a date that has already expired

```
<?php
 setcookie( "name", "", time()- 60, "/","", 0);
 setcookie( "age", "", time()- 60, "/","", 0);
?>
<html><head> <title>Deleting Cookies with PHP</title> </head>
<body>
<?php echo "Deleted Cookies" ?> </body></html>
```

REGULAR EXPRESSIONS IN PHP

Regular expressions are nothing more than a sequence or pattern of characters. They provide the foundation for pattern-matching functionality.

PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression: POSIX Regular Expressions and PERL Style Regular Expressions.

POSIX Regular Expressions

- The structure of a POSIX regular expression is similar to a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.
- The simplest regular expression is one that matches a single character.
  - **Brackets:** Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.
  - **Quantifiers:** The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character having a specific connotation. The +, *, ?, {int. range}, and $ flags all follow a character sequence.
  - **Predefined Character Ranges**: For programming convenience several predefined character ranges, also known as character classes, are available. Character classes specify an entire range of characters, for example, the alphabet or an integer set.

Expression	Description
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.
p+	It matches any string containing at least one p.
p*	It matches any string containing zero or more p's.
p?	It matches any string containing zero or more p's. This is just an alternative way to use p*.
p{N}	It matches any string containing a sequence of N p's
p{2,3}	It matches any string containing a sequence of two or three p's.

PHP currently offers seven functions for searching strings using POSIX-style regular expressions:

Function	Description
ereg()	The ereg() function searches a string specified by string for a string specified by pattern, returning true if the pattern is found, and false otherwise.
ereg_replace()	The ereg_replace() function searches for string specified by pattern and replaces pattern with replacement if found.
eregi()	The eregi() function searches throughout a string specified by pattern for a string specified by string. The search is not case sensitive.
eregi_replace()	The eregi_replace() function operates exactly like ereg_replace(), except that the search for pattern in string is not case sensitive.
split()	The split() function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.
spliti()	The spliti() function operates exactly in the same manner as its sibling split(), except that it is not case sensitive.
sql_regcase()	The sql_regcase() function can be thought of as a utility function, converting each character in the input parameter string into a bracketed expression containing two characters.

PERL Style Regular Expressions:

Perl-style regular expressions are similar to their POSIX counterparts. The POSIX syntax can be used almost interchangeably with the Perl-style regular expression functions

- **Metacharacters:** A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

Metacharacters	Description
.	a single character
\s	a whitespace character (space, tab, newline)
\S	non-whitespace character
\d	a digit (0-9)
\D	a non-digit

\w	a word character (a-z, A-Z, 0-9, _)
\W	a non-word character
[aeiou]	matches a single character in the given set
[^aeiou]	matches a single character outside the given set
(foo|bar|baz)	matches any of the alternatives specified

Modifiers: Several modifiers are available that can make the user work with regexps much easier, like case sensitivity, searching in multiple lines etc.

Modifiers	Description
I	Makes the match case insensitive
M	Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of string boundary
O	Evaluates the expression only once
S	Allows use of . to match a newline character
X	Allows the user to use white space in the expression for clarity
G	Globally finds all matches
Cg	Allows a search to continue even after a global match fails

PHP's Regexp PERL Compatible Functions

Functions	Description
preg_match()	The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
preg_match_all()	The preg_match_all() function matches all occurrences of pattern in string.
preg_replace()	The preg_replace() function operates just like ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.
preg_split()	The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.
preg_grep()	The preg_grep() function searches all elements of input_array, returning all elements matching the regexp pattern.
preg_ quote()	Quote regular expression characters