Date:2021-01-28

* Word censoring:
  + Continue of find and replace for getting a data from user and perform find and replace on it.
  + str\_ireplace(); - used for ignore a case sensitivity.
  + Example:
    - <?php
    - if( isset($\_POST['u\_input'] ) && !empty($\_POST['u\_input'])) {
    - echo "<center>";
    - $str = $\_POST['u\_input'];
    - $find = array("Hello" ,"Student" ,"Marwadi university");
    - $replace = array("Hi", "Intern" ,"Cybercom Creation");
    - echo $new\_str = str\_ireplace($find, $replace, $str);
    - echo "</center>";
    - }
    - ?>
    - <!DOCTYPE html>
    - <html>
    - <head>
    - <title>Find and replace text</title>
    - </head>
    - <body>
    - <center><hr>
    - <form action="#" method="post">
    - <textarea rows="6" cols="30" name="u\_input">
    - <?php
    - if (isset($str)) {
    - echo $str;
    - }
    - ?>
    - </textarea><br><br>
    - <input type="submit" value="Find and replace">
    - </form>
    - </center>
    - </body>
    - </html>
* Find and replace application:
  + Example:
    - <?php
    - if( isset($\_POST['u\_input'] ) && isset($\_POST['search']) && isset($\_POST['replace'])) {
    - $data = $\_POST['u\_input'];
    - $search = $\_POST['search'];
    - $replace = $\_POST['replace'];
    - $search\_len = strlen($replace);
    - $data1 = '';
    - $data2 = '';
    - $offset = 0;
    - if (!empty($data) && !empty($search) && !empty($replace)) {
    - while ($str\_pos = strpos($data, $search ,$offset)) {
    - $offset = $str\_pos + $search\_len;
    - $data = substr\_replace($data, $replace, $str\_pos ,$search\_len);
    - }
    - echo "<center>";
    - echo $data;
    - echo "</center>";
    - } else {
    - echo "fill data properly...";
    - }
    - $data1 = $\_POST['u\_input'];
    - echo "<center><b>Another way with str\_replace()</b><br>";
    - echo $data2 = str\_replace($search, $replace, $data1);
    - }
    - ?>
    - <!DOCTYPE html>
    - <html>
    - <head>
    - <title>Find and replace Application</title>
    - </head>
    - <body>
    - <center><hr>
    - <form action="#" method="post">
    - <textarea rows="6" cols="30" name="u\_input">
    - <?php
    - if (isset($str)) {
    - echo $str;
    - }
    - ?>
    - </textarea><br><br>
    - <b>Search For :</b><br>
    - <input type="text" name="search"><br><br>
    - <b>Replace With :</b><br>
    - <input type="text" name="replace"><br><br>
    - <input type="submit" value="Find and replace">
    - </form>
    - </center>
    - </body>
    - </html>
* Time stamps:
  + time(): return a time in second from a 1970.
  + date(‘formate ’,time in second):return a time and date as we want
    - Example:
      * <?php
      * //time() return a epoch time
      * echo $time = time();
      * //date()
      * echo "<br>".date('H : i : s',$time);
      * //date with month
      * echo "<br>".date('Y - m - d @ H : i : s')
      * ?>
* Modifying timestamps:
  + strtotime(+1 week day etc) – add a time in current date.
* Random number generator:
  + rand() -return a random number.
  + rand(min,max) – return a random no. bet min and max.
  + Example:
    - <?php
    - {
    - $rand = rand(1,6);
    - echo "you roll a dice $rand.";
    - }
    - ?>
    - <hr>
    - <form action="#" method="post">
    - <input type="submit" name="roll" value="Roll a dice">
    - </form>
* $\_SERVER[]:
  + Attributes:
    - $\_SERVER['PHP\_SELF'] Returns the filename of the currently executing script
    - $\_SERVER['GATEWAY\_INTERFACE'] Returns the version of the Common Gateway Interface (CGI) the server is using
    - $\_SERVER['SERVER\_ADDR'] Returns the IP address of the host server
    - $\_SERVER['SERVER\_NAME'] Returns the name of the host server
    - $\_SERVER['SERVER\_SOFTWARE'] Returns the server identification string (such as Apache/2.2.24)
    - $\_SERVER['SERVER\_PROTOCOL'] Returns the name and revision of the information protocol (such as HTTP/1.1)
    - $\_SERVER['REQUEST\_METHOD'] Returns the request method used to access the page (such as POST)
    - $\_SERVER['REQUEST\_TIME'] Returns the timestamp of the start of the request (such as 1377687496)
    - $\_SERVER['QUERY\_STRING'] Returns the query string if the page is accessed via a query string
    - $\_SERVER['HTTP\_ACCEPT'] Returns the Accept header from the current request
    - $\_SERVER['HTTP\_ACCEPT\_CHARSET'] Returns the Accept\_Charset header from the current request (such as utf-8,ISO-8859-1)
    - $\_SERVER['HTTP\_HOST'] Returns the Host header from the current request
    - $\_SERVER['HTTP\_REFERER'] Returns the complete URL of the current page (not reliable because not all user-agents support it)
    - $\_SERVER['HTTPS'] Is the script queried through a secure HTTP protocol
    - $\_SERVER['REMOTE\_ADDR'] Returns the IP address from where the user is viewing the current page
    - $\_SERVER['REMOTE\_HOST'] Returns the Host name from where the user is viewing the current page
    - $\_SERVER['REMOTE\_PORT'] Returns the port being used on the user's machine to communicate with the web server
    - $\_SERVER['SCRIPT\_FILENAME'] Returns the absolute pathname of the currently executing script
    - $\_SERVER['SERVER\_ADMIN'] Returns the value given to the SERVER\_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
    - $\_SERVER['SERVER\_PORT'] Returns the port on the server machine being used by the web server for communication (such as 80)
    - $\_SERVER['SERVER\_SIGNATURE'] Returns the server version and virtual host name which are added to server-generated pages
    - $\_SERVER['PATH\_TRANSLATED'] Returns the file system based path to the current script
    - $\_SERVER['SCRIPT\_NAME'] Returns the path of the current script
    - $\_SERVER['SCRIPT\_URI'] Returns the URI of the current page
* header():
  + used for update a header info.
  + Header(location :’filr name’)
* ob\_start():
  + it used for start a buffer output.
  + ob\_end\_flush() – it clean the buffer output.
* Getting a visitors IP address:
  + $\_SERVER[‘REMOTE\_ADDR’]; - used for getting a user ip address.
* Better way to detect an ip address:
  + Some time we cant get a right ip address of user due to a proxy server then we can use this method.
  + $\_SERVER['HTTP\_CLIENT\_IP']
  + $\_SERVER['HTTP\_X\_FORWARDED\_FOR']
* Detecting a visitors browser:
  + $\_SERVER['HTTP\_USER\_AGENT'];
  + $browser = get\_browser();
* $\_GET[]:
  + Used to get a value from a form.
  + Ex:
    - <?php
    - if(isset($\_GET['day']) && isset($\_GET['month']) && isset($\_GET['year'])){
    - $day = $\_GET['day'];
    - $month = $\_GET['month'];
    - $year = $\_GET['year'];
    - if (!empty($day) && !empty($month) && !empty($year)) {
    - echo $day;
    - echo "<br>$month";
    - echo "<br>$year";
    - } else {
    - echo "Fill all fileds.....";
    - }
    - }
    - ?>
    - <form action="#" method="get">
    - <b>DAY:</b><br>
    - <input type="text" name="day"><br><br>
    - <b>MONTH:</b><br>
    - <input type="text" name="month"><br><br>
    - <b>YEAR:</b><br>
    - <input type="text" name="year"><br><br>
    - <input type="submit" name="submit">
    - </form>
* $\_POST[]:
  + Used for a big info. And sensitive info. Like password.
  + Because when we submit form info. Not display on a url.
  + Ex:
    - <?php
    - $match = "pass123";
    - if(isset($\_POST['pass'])){
    - $pass = $\_POST['pass'];
    - if (!empty($pass)) {
    - if ($match == $pass) {
    - echo "Password match....";
    - } else {
    - echo "Password not match...";
    - }
    - } else {
    - echo "Enter Password.....";
    - }
    - }
    - ?>
    - <form action="#" method="post">
    - <b>Password:</b><br>
    - <input type="password" name="pass"><br><br>
    - <input type="submit" name="submit">
    - </form>