❖ **Word censoring (find and replace words using str_replace( ))**

- Today the practice started from word censoring in which we created a text area field.
- Then if the word which we want to change is available in that text area then we have to replace those word or words with some another word.
- It can be done by str_replace() function . but the problem is when there are capital letters then It will not work.
- So there is a function as solution named **str_ireplace().** it will search and replace text in case insensitive manner.
- Example :

```
<?php
   $find = ['sumant','ankit','mayur'];
   $replace = ['Su***t','A***t','M***r'];
   if(isset($_POST['user_input']) && !empty($_POST['user_input'])) {
      $user_input = $_POST['user_input'];
 $new_user_input = str_ireplace($find , $replace , $user_input);
      echo $new_user_input;
   }
?>


<!DOCTYPE html>
<html lang="en">
<head>
   <title>Word censoring</title>
</head>
<body>
   <hr>
   <form action="php_word_censoring.php" method="post">
      <textarea name="user_input" rows="6" cols="30">
      <?php if(isset($_POST['user_input'])){echo $user_input;}
```

```
        ?></textarea> <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

❖ **Small search and replace application using string function**

- In this application there is one text area field and two text fields. First we have to enter some text in text area and then one word in search and one word in replace text field
- It will search the word and replace all occurrences of that word
- Example :

```
<?php
    $offset = 0;
    if(isset($_POST['user_input']) && isset($_POST['search_for']) &&
isset($_POST['replace_with'])) {
        $user_input = $_POST['user_input'];
        $find = $_POST['search_for'];
        $replace = $_POST['replace_with'];

        $length = strlen($find);

        if(!empty($user_input) && !empty($find) && !empty($replace)) {
            while($str_pos = strpos($user_input , $find , $offset)) {
                $offset = $str_pos + $length;
                $user_input = substr_replace($user_input , $replace , $str_pos ,
$length);
            }
            echo $user_input;
        }
```

```
        else {
            echo 'please fill in all the fields';
        }
    }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Find and Replace Application</title>
</head>
<body>
    <hr>
    <form action="php_find_and_replace_application.php" method="post">
        <textarea name="user_input" rows="6" cols="30"></textarea><br><br>

        <label for="search_for">Search for : </label> <br>
        <input type="text" name="search_for" id="search-for"> <br><br>

        <label for="replace_with">Replace With : </label> <br>
        <input type="text" name="replace_with" id="replace_with"> <br><br>

        <input type="submit" value="Find and Replace">
    </form>
</body>
</html>
```

❖ **Timestamps**

- Timestamps are very useful to know the date and the time
- We can get the time by using **time().** It will return time In seconds since 1st january, 1970 (Epoch time)
- We can also do formatting to it using **date()** function
- In date () we have to pass two arguments , first is format and the second is time().
- By default the time() will display the time of berlin because by default berlin's time zone is set'
- To set the time zone of Asia/Calcutta we have to use a function called **date_default_timezone_set('Asia/Calcutta');**
- We can also modify the date and time too. We can modify it in two ways
- First way is by adding numeric values to time() function like , time() + (60*60*5) this will add 5 days to current time
- Second way of modifying time is **strtotime()**
- Strtotime() function allows us to modify time in string manner. We just have to write in simple string like , date('format' , strtotime('+1 day 2 hours));
- It will increase 1 day and 2 hours in date time.
- Let's see a complete example of timestamp
- Example :

```
<?php
   include 'php_starter.php';
   $sub_heading = 'Modify timestamp';
   echo '<h2>'.$sub_heading.'</h2>';
   date_default_timezone_set('Asia/Calcutta');

   $time = time();
   echo '<br>current date/time : '. date('D M Y @ H:i:s',$time);
   echo '<br>current date/time : '. date('d m y @ h:i:s',$time);

   // add 1 day
   echo '<br><br>Added one day into current time : '. date('d M Y @ H:i:s',
strtotime('+1 day'));
   // add 1 week
```

```
    echo '<br><br>Added one week into current time : '. date('d M Y @ H:i:s', strtotime('+1 week'));

    // add 1 hour

    echo '<br><br>Added one hour into current time : '. date('d M Y @ H:i:s', strtotime('+1 hour'));

    // one month, 2 hours , 25 minutes , 30 seconds

    echo '<br><br>Added one month, 2 hours , 25 minutes , 30 seconds into current time : '. date('d M Y @ H:i:s', strtotime('+1 month 2 hours 25 minutes 30 seconds'));

    // one month, 2 hours , 25 minutes , 30 seconds

    echo '<br><br>added 5 hours : '. date('d M Y @ H:i:s' , $time + (60*60*5));

?>
```

❖ **Random Numbers**

- **rand()** is used to generate the random numbers in php.
- it will give us a random number whenever the page is refreshed
- there is another method called **getrandmax()** it will give us the maximum number up to which the rand() can generate random numbers.
- Example :

```
<?php

    include 'php_starter.php';

    $sub_heading = 'Generate Random Number';

    echo '<h2>'.$sub_heading.'</h2>';


    // getrandmax() is used to get the maximum number can be generated using rand()


    if(isset($_POST['roll'])) {

        $dice  = rand(1, 6);

        echo 'you rolled a : ' . $dice .'<br><br>';

    }
```

```
?>


<!DOCTYPE html>

<html lang="en">

<body>

  <form action="" method="post">

    <input type="submit" name="roll" value="Roll the dice">

  </form>

</body>

</html>
```

❖ **$_SERVER (super global variable)**

- **$_SERVER** is a super global variable.
- all the super global variables are available in all the scopes
- $_SERVER variable holds the information about headers , paths and script locations.
- We can get so many variables information from $_SERVER variable. Some important information that we can get from it is as follows
  - $_SERVER[‘SCRIPT_NAME] :- returns the path of current script
  - $_SERVER[‘HTTP_HOST’] :- return the host header from the current request
  - $_SERVER[‘SERVER_ADDR’] :- return ip address of host server
  - $_SERVER[‘REMOTE_ADDRESS’] :- return the ip address from where the user is viewing this page
- Example :

```
<?php

  echo '<br><br>'. $_SERVER['HTTP_HOST'];

  echo '<br><br>'. $_SERVER['SCRIPT_NAME'];

  echo '<br><br>'. $_SERVER['SERVER_ADDR'];

?>
```

- We can get the ip address of user using $_SERVER variable.
- There is a a way or we can say good way to get ip address of users which is displayed in a following example

- Example :

```
<?php
function getIpAddress() {
   // internet ip address
   if(!empty($_SERVER['HTTP_CLIENT_IP'])) {
      $ip_address = $_SERVER['HTTP_CLIENT_IP'];
   }
   // if users is using proxy
   else if(!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
      $ip_address = $_SERVER['HTTP_X_FORWARDED_FOR'];
   }
   // local or remote address
   else {
      $ip_address = $_SERVER['REMOTE_ADDR'];
   }
   return $ip_address;
}
echo 'ip adress is - '. getIPAddress();
?>
```

- When we want to provide different value to action attribute of form tag at that time we can use $_SERVER[‘script_name’].
- It is very useful when the same form is used in multiple pages
- Example :
  Php_script_variable.php :

```
<?php $server_script = $_SERVER['SCRIPT_NAME']; ?>

<form action="<?php echo $server_script; ?>" method="POST">
   <input type="submit" value="Submit" name="click">
</form>
```

Process1.php :

```
<?php
   include 'php_script_variable.php';

   if(isset($_POST['click'])) {
   echo 'process1';
  }
?>
```

Process2.php :

```
<?php
 include 'php_script_variable.php';

  if(isset($_POST['click'])) {
  echo 'process2';
}
?>
```

❖ **header()**

- header() is used to send a raw HTTP header. See the » HTTP/1.1 specification for more information on HTTP headers.
- Parameters of header() are as follows :
- **1) header string**
    - There are two special-case header calls. The first is a header that starts with the string "HTTP/" (case is not significant), which will be used to figure out the HTTP status code to send.
    - The second special case is the "Location:" header. Using this we can send or redirect user to another url or page

- **2) replace**
    - The optional replace parameter indicates whether the header should replace a previous similar header, or add a second header of the same type. By default it will replace, but if you pass in false as the second argument you can force multiple headers of the same type.
- **3) response code**
    - Forces the HTTP response code to the specified value. Note that this parameter only has an effect if the header is not empty.
- Example of header() with header string argument only :

```
<?php
   $redirec_path = 'https://google.co.in';
   $redirect = true;
   if($redirect == true) {
      header('Location: '.$redirec_path);
   }
?>
```

❖ **Ob_start() & ob_end_clean()**

- The ob_start() function creates an output buffer.
- A callback function can be passed in to do processing on the contents of the buffer before it gets flushed from the buffer.
- **ob_end_clean()** function will clean all the data and will be not sent to the browser.
- Example :

```
<html>
<body>
<?php
ob_start();
echo "This content will not be sent to the browser.";
ob_end_clean();
```

```
echo "This content will be sent to the browser.";
?>
</body>
</html>
```

- **$_POST (super global variable)**

- $_POST is a super global variable in php. We can use $_POST variable anywhere we want to use.
- It is mainly used to get all the data of form which is sent through **method= "post"**.
- For example suppose we have a form with post method and if the user submit the form and send the data then to get or to read that data we can use $_POST and we can also store those data into variables and then use those variables to perform any kind of operations
- Let's understand this by an example
- Example :

```
<?php
   $pass = 'pass8320247191';
   if(isset($_POST['password'])) {
      $password = $_POST['password'];

      if(!empty($password)) {
         if($password===$pass) {
            echo 'password is correct. your welcome';
         }
         else {
            echo 'please enter correct password';
         }
      }
      else {
         echo 'please enter the password';
```

```
    }
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <title>PHP</title>
</head>
<body>
  <form action="" method="post">
    <label for="password">Password</label><br>
    <input type="password" name="password" id="password"> <br><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

❖ **$_GET (super global variable)**

- $_GET is super global variable in php.
- It is mainly used to get all the data of form which is sent through **method= "get"**.
- It can also collect data that is sent from url
- Example :

```
<?php
  if(isset($_GET['first_name']) && isset($_GET['middle_name']) &&
isset($_GET['last_name'] )) {
    $first_name = $_GET['first_name'];
    $middle_name = $_GET['middle_name'];
```

```
    $last_name = $_GET['last_name'];


    if(!empty($first_name) && !empty($middle_name) &&
!empty($last_name)) {
        echo 'Full name is : ' . $first_name . ' ' . $middle_name . ' ' .
$last_name;
    }
    else {
        echo 'please fill in all the fields';
    }
  }
?>


<!DOCTYPE html>
<html lang="en">
<head>
  <title>PHP</title>
</head>
<body>
  <form action="" method="get">
    <label for="first_name">First Name</label><br>
    <input type="text" name="first_name" id="first_name"> <br><br>


    <label for="middle_name">Middle Name</label><br>
    <input type="text" name="middle_name" id="middle_name"> <br><br>


    <label for="last_name">Last Name</label><br>
    <input type="text" name="last_name" id="last_name"> <br><br>
```

```
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```