UNIT: 4 Forms Handling, Session, Cookies

4.1. Form controls: Text Box, Textarea, List Box, Dropdown, Check Box, Radio Box, Buttons, Upload, color, date etc.

Introduction to Forms:

- ✓ Whenever you are going to design a page, which can accept information from users through various input fields.
- ✓ We can define a form in PHP page using <form>..</form> tag.
- ✓ ***A form can contain input elements like***
    1. Textfield
    2. Textarea
    3. Checkboxes
    4. Radio-buttons
    5. Drop-down list
    6. Submit buttons etc..
- ✓ *Forms are used to pass data to a server.*
- ✓ *The FORM element has no formatting attributes.*

Input Elements: Input elements are used to get input from user in various format. Its properties are specified in TYPE attribute of <INPUT> ..</INPUT> tag.

Input Elements Properties:

Sr No	**Type**	**Control**
1	Text	Textbox
2	Hidden	Hidden box
3	Password	Textbox with password
4	Submit	Submit button
5	Checkbox	Check box
6	Radio	Radio button

- ✓ **TYPE:** Type of INPUT entry field
- ✓ **NAME:** Variable name passed to application.
- ✓ **VALUE:** Data associated with variable name
- ✓ **CHECKED:** Botton/box checked by default
- ✓ **SIZE:** Number of characters in textfield
- ✓ **MAXLENGTH:** Maximum number of characters accepted

1. **Using Textbox:** User can enter text or single line using textbox

```
Syntax:<input type="text" name="Textname" value=”defaultvalue”>
```

```
Example:
<form>
    Enter your Name:<input type="text" name="txtName">
</form>
```

2. **Using Password:** User can enter confidential information such as password, bank acountno etc.

```
Syntax: <input type="password" name="txtPassword" value=”defaultvalue”>
Example:
<form>
   Enter your Password: <input type="password" name="txtPassword">
</form>
```

3. **Using Textarea:**User can enter multiple line of text like address, feedback, comments etc.

```
Syntax: <textarea name="txtname" rows=”rowsize” cols=”columnsize”>   </textarea>
Example:
<form>
   Enter Address:
       <textarea name="txtarea1" rows=”5” cols=”20”>
       </textarea>
</form>
```

4. Using Checkbox

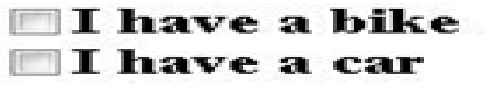
- ✓ It allows you to represent list of options to the users from which user can select none, one or more than one options at a time.
- ✓ Thus it useful when you want to represent various choices to user from which he/she can select choices as per his/her requirement.
- ✓ It displays as small square on web page

Syntax : <input type=”checkbox” name=”Name” value=”Value” [checked]> Text </input>

TYPE: Indicates type of input element. So it is checkbox.
NAME: Variable name passed to application.
VALUE: Data associated with variable name
CHECKED: Checkbox checked by default

Example:

```
<form>
<input type="checkbox" name="vehicle" value="Bike">  I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">   I have a car
</form>
```


I have a bike
I have a car

5. Using Radio button

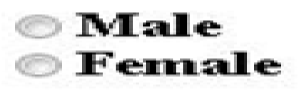
- ✓ It allows you to represent list of options to the users from which user can select only one options at a time.
- ✓ Thus it useful when you want to represent various choices to user from which he/she can select only one choice as per his/her requirement.
- ✓ It displays as small circle on web page

Syntax : <input type=”radio” name=”Name” value=”Value” [checked]> Text </input>

TYPE: Indicates type of input element. So it is checkbox.
NAME: Variable name passed to application.
VALUE: Data associated with variable name
CHECKED: Checkbox checked by default

Example

```
<form>
<input type="radio" name="gender" value="male">Male<br>
<input type="radio" name="gender" value="female">Female
 </form>
```

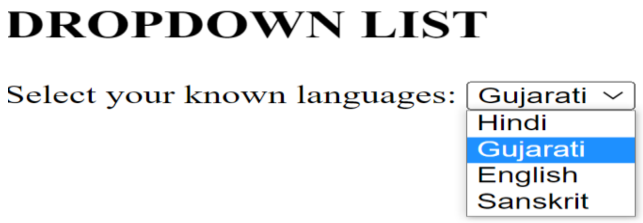


6. Dropdown list and Listbox:

- ✓ For Dropdown list and Listbox control, <SELECT></SELECT> element is used , where the attributes are set differently.
- ✓ The list items are added to <SELECT> elements by <OPTION></OPTION> elements.
- ✓ Select element's attributes are:
    - o NAME: Name of variable to be sent to application
    - o SIZE: Sets the number of visible choices.
    - o MULTIPLE: User can make multiple selections. By default one is allowed.
- ✓ Option element's attribute is:
    - o SELECTED: If it is selected when the document is initially loaded. It is an error for more than one option to be selected.

Dropdown list: This is like combobox control of Visual basic. Dropdown list is used to select one option from given list of choice.

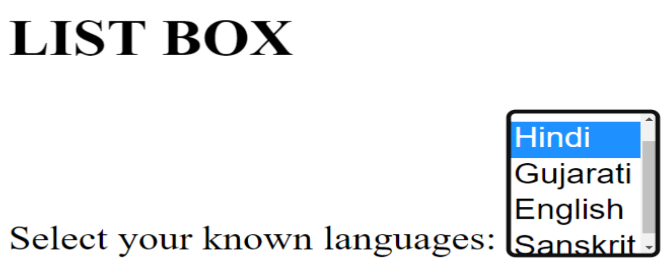
```
<form>
Select your known languages:
<select name="languages">
<option selected>Hindi</option>
<option>Gujarati</option>
<option>English</option>
<option>Sanskrit</option>
</select>
</form>
```



List Box: This is like List control of visual basic. Select list is used to select one or more option from given list of choice.

```
<html>
<body>
<h2>LIST BOX</h2>
<form>
Select your known languages:
<select name="select" size="4" multiple="true">
```

```
<option></option>
<option selected>Hindi</option>
<option>Gujarati</option>
<option>English</option>
<option>Sanskrit</option>
</select>
</form>
</body></html>
```


LIST BOX

Select your known languages:

7. Button control:

Push Button: This element would be used to cause an action to take place.
Syntax: <input type=”button” name=”button” value=”button”>

- **TYPE:** Indicates type of input element.
- **NAME:** Variable name passed to application.
- **VALUE:** Data associated with variable name

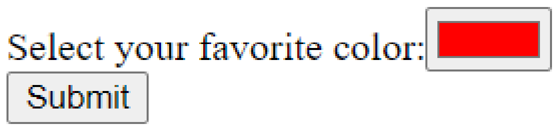
Submit Button: Every set of form tags requires a Submit button. It causes the browser to send the names and values of other elements to application specified by ACTION attribute of Form element.
Syntax: <input type=”submit” name=”submit” value=”submit”>

- **TYPE:** Indicates type of input element.
- **NAME:** Variable name passed to application.
- **VALUE:** Data associated with variable name

Image Submit Button: Allows you to substitute an image for the standard submit button.
Syntax: <input type=”image” src=”url” name=”image”>
Reset Button: It is good idea to include one of these for each form where user are entering data. It allows user to clear all input in form.
Syntax: <input type=”reset” name=”reset” value=”reset”>>

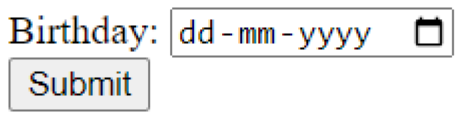
7. color : It defines a color picker. The default value is #000000 (black).

Syntax: <input type="color">

<form > Select your favorite color:<input type="color" name="favcolor" value="#ff0000"> <input type="submit"> </form>	 Select your favorite color: 🟥

8. Date : It defines a date picker. The resulting value includes the year, month, and day

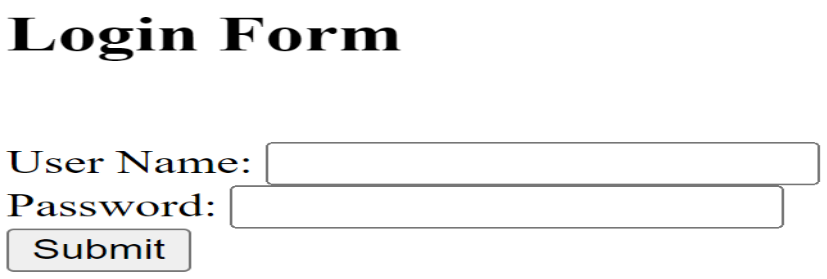
Syntax: <input type="date">

`<form>` `Birthday:<input type="date" name="birthday"><br>` `<input type="submit">` `</form>`	 Birthday: dd-mm-yyyy Submit

Creating the form: Following code is used to create a Login Form

```
<html>
<body>
<form>
<h2>Login Form</h2><br>
User Name: <input type="text" size="20" ><br>
Password:   <input type="text" size="20"><br>
<input type="submit" name="submit" value="Submit">
</form>
</body>
```

Output



Login Form

User Name:
Password:

```
</html>
```

Attributes of Forms:*[ACTION and METHOD IMP]*

- **ACTION:** It is the URL of the CGI (Common Gateway Interface) program that is going to accept data from the form, Process it, then Send a response back to browser.
- **METHOD:GET(default) or POST** specifies which HTTP method will be used to send the form's contents to web server.
- **ENCTYPE:** Mechanism used to encode the form contents. You can leave this attribute as default.
- **NAME:** This attribute specifies the name of the Input control.
- **TARGET:** Target frame where response page will show up.

4.2. Retrieving form data using GET and POST methods

✓ When a form is submitted to a PHP script, information from that form is automatically made available to script.
✓ There are many ways to access this information here explain GET and POST method of form object.

GET Method:

- The GET method passes arguments from in page to next page as a part of the URL Query String.
- When used for form handling, GET appends the indicated variable name and value to the URL designated in the ACTION attribute with a question mark separator.

- Each item submitted via GET method is accessed in the handler via $_GET array.
- Example: form1.php

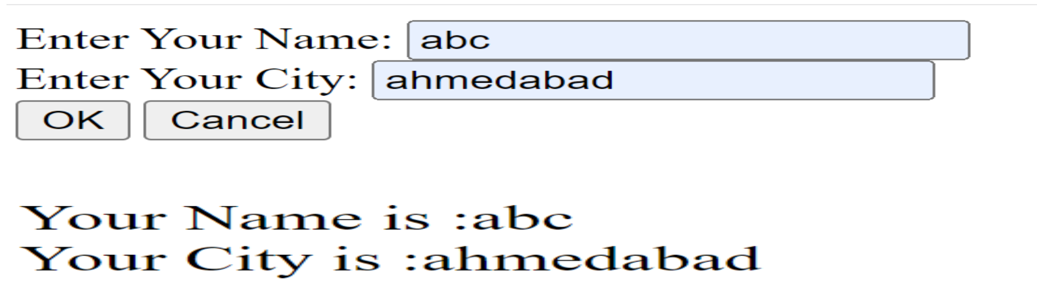
```
<html>
<body>
<form action="form2.php" method="GET">
Enter Your Name: <input type="text"name="name"><br>
Enter Your City: <input type="text" name="city" ><br>
<input type="submit" value="OK" >
<input type="reset" value="Cancel" >
</form>
</body>
</html>
```

URL: http://localhost/ami/form2.php?name=abc&city=ahmedabad

The “form2.php” file looks like this:

```
<?php
$Name=$_GET['name'];
$City=$_GET['city'];
echo "Your Name is :". $Name."<br>";
echo "Your City is :". $City."<br>";
?>
```

Output:



Enter Your Name: abc
Enter Your City: ahmedabad
OK Cancel

Your Name is :abc
Your City is :ahmedabad

Advantages of GET method:

It constructs an actual new and differentiable URL query string so user can bookmark this page.

Disadvantages of GET method:

- It is not suitable for login form because username & password fully visible onscreen.
- Every GET submission is recorded in the web server log, data set included.
- The length of URL is limited so limited data pass using GET method.
  (Query string to be limited 255 characters.)

POST Method:

- POST method is the preferred method of form submission.
- The form data set is included in the body of the form when it is forwarded to the processing agent (web server).
- No visible change to the URL will result according to the different data submitted.
- Each item submitted via POST method is accessed in the handler via the $_POST array.

Advantages of POST method:

- It is more secure then GET because user entered information is never visible in URL.
- There is much larger limit on the amount of data that can be passed (a couple of kilobytes).

Disadvantages of POST method:

- The result at a given moment cannot be book marked.
- The result should be expired by the browser, so that an error will result if the user employs the Back button to revisit the page.
- This method can be incompatible with certain firewall setups

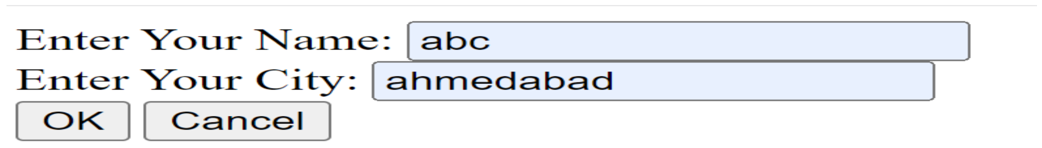
Example: form1.php

```
<html>
<body>
<form action="form2.php" method="POST">
Enter Your Name: <input type="text"name="name"><br>
Enter Your City: <input type="text" name="city" ><br>
<input type="submit" value="OK" >
<input type="reset" value="Cancel" >
</form>
</body>
</html>
```

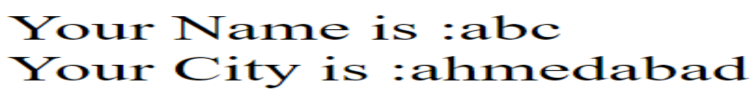
URL: http://localhost/form2.php

```
<?php
$Name=$_POST['name'];
$City=$_POST['city'];
echo "Your Name is :". $Name."<br>";
echo "Your City is :". $City."<br>";
?>
```

Output:



```
Enter Your Name: abc
Enter Your City: ahmedabad
```



```
Your Name is :abc
Your City is :ahmedabad
```

✓ ***One can also use $_REQUEST['variableName'] instead of $_POST['variableName'] or $_GET['variableName']***

DIFFERENCE BETWEEN GET AND POST:

GET Method	**POST Method**

Using GET method data is sent from one page to other in the URL	Using POST method data is sent from one page to other within the body of the HTTP request
The GET method, appends name/value pairs to the URL	POST method packages the name/value pair inside body of HTTP request, which makes for a clean URL
The length of URL is limited, so it works if there are few parameters.	POST method imposes no size limitations on forms output.
It is insecure because parameters passed on the URL are visible in address field of browser.	It is secure because submitted data are passed through HTTP handler.
It can't be used to send binary data, like images or word documents, to server.	Using POST method can be used to send ASCII as well as binary data. The data send by POST method can be accessed using $_POST superglobal variable

4.3. Form Validation using PHP

While designing the form that accepts input from user, it is required to test input values entered bu the user such as:

- ✓ Whether it is in required format or not
- ✓ Whether value entered or left by user

This process is known as Validation.

Regular Expression: It is a pattern that is used to match various text strings.

- ✓ It is used to validate user input to determine whether it is entered in predefined format or not.
- ✓ Using Regular Expression, you can validate numbers, email address, date, contact no etc.
- ✓ Regular Expression use ereg() function or eregi() function to validate user input.
- ✓ ereg() function is case sensitive and eregi() function is case insensitive.

  Syntax for ereg() function:

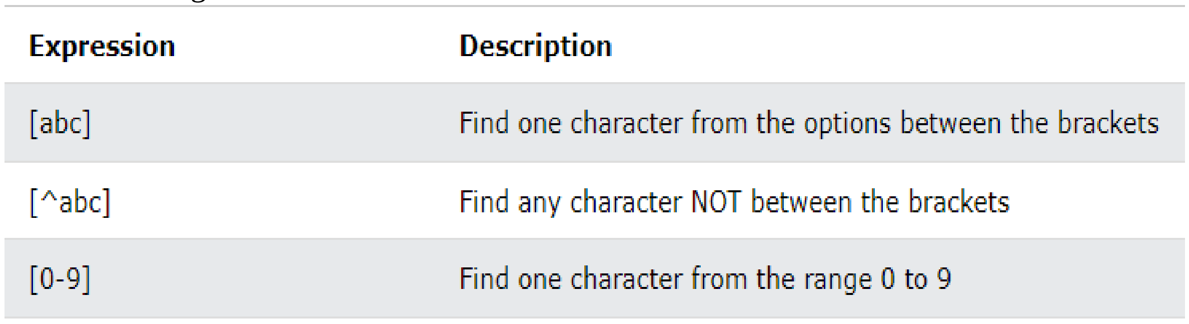
  ereg($pattern,$string)

  where, $pattern specifies the **pattern** to be match

  $string specifies the string to be matched with pattern.

Metacharacters used in PHP

- ✓ Brackets are used to find a range of characters and Metacharacters are characters with a special meaning:

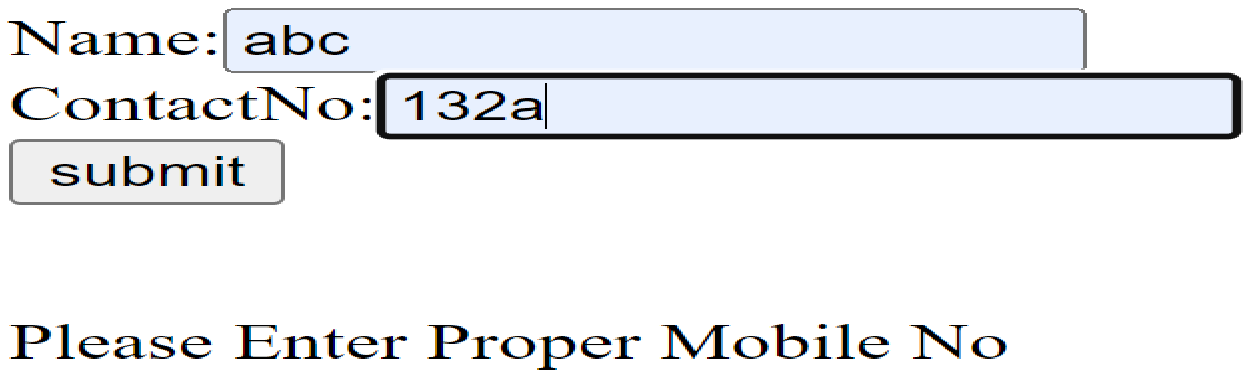


Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

```
<html>
```

```
<head></head>
<body>
<form method="POST" >
Name:<input type="text" name="txtName"><br>
ContactNo:<input type="text" name="txtNo"><br>
<input type="submit" name="submit" value="submit">
</form>
</body>
</html>

<?php
if (isset($_POST['submit']))
{
$Name=$_POST['txtName'];
$ContactNo=$_POST['txtNo'];

if(empty($Name))
{
echo "Please Enter Name";

}

if(empty($ContactNo))
{
echo "Please Enter Contact No";
}

if(!preg_match("/[0-9]{10}/",$ContactNo))
{
echo "<br>Please Enter Proper Mobile No";
}
}
?>
```

Output:


Name: abc
ContactNo: 132a

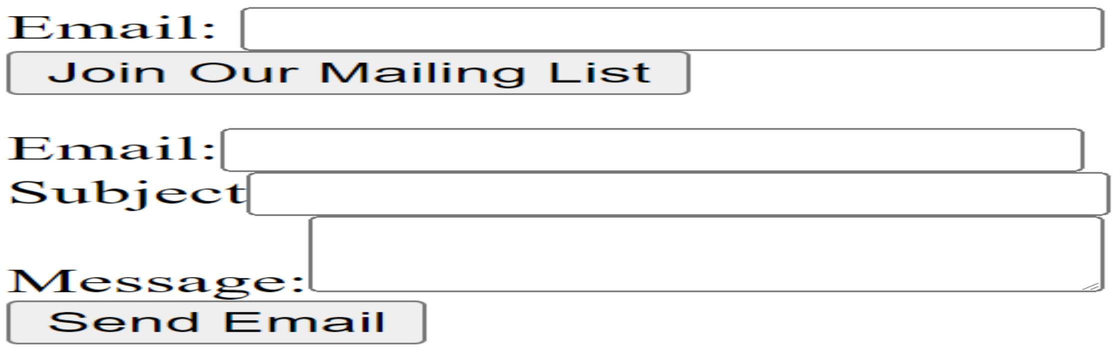
```
Please Enter Proper Mobile No
```

4.4. Working with multiple forms

i. A web page having multiple forms

File name: multiple.php

```
<form name="mail" method="POST" action="multiple1.php">
Email: <input type="text" name="email" ><br>
<input type="submit" name="msubmit" value="Join Our Mailing List" />
</form>

<form name="contactus" method="post" action="multiple1.php">
Email:<input type="text" name="email" ><br>
Subject<input type="text" name="subjet"><br>
Message:<textarea name="message"></textarea><br>
<input type="submit" name="csubmit" value="Send Email" />
</form>
```



ii. A form having multiple submit buttons

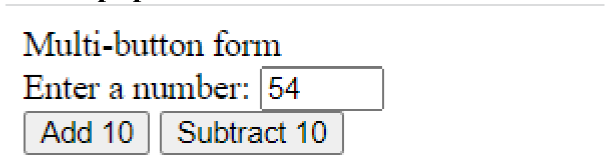
a.php

```
<html>
<head>Multi-button form</head>
<body>

<form action="b.php" method="post">
 Enter a number: <input type="text"
name="number" size="3"> <br>
<input type="submit" name="add"
value="Add 10">
<input type="submit" name="subtract"
value="Subtract 10"> </form>

</body>
</html>
```

b.php

```
<?php
if ($_POST['add'])
{
   echo $_POST['number']+10;
}
 else if ($_POST['subtract'])
{
   echo $_POST['number']-10;
}

?>
```

Output:
Run a.php



64

4.5. Session: creating a session, storing and accessing session data and destroying session

What is a PHP Session?

Definition:A session is a way to store information (in variables) to be used across multiple pages.

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

3. Passing variables through session variables:

- An alternative way to make data accessible across various pages of an entire website is to use a PHP session.
- A **session** creates a file in a temporary directory on the **server** where registered session variables and their values are stored.
- Location of temporary file is determined by a setting in the php.ini file called session.save_path.
- To start a session call function session_start(). This function first checks if a session is already started and if none is started then it starts one.
- Session variables are stored in associative array called $_SESSION[].
- **Example1: Demonstrate basic session behaviour**
  Code: session.php
```
<?php
        session_start();
        $_SESSION['username']='abc';
        $_SESSION['password']='123';
?>
```
- To display the username code looks like this:
  Code:session2.php
```
<?php
        session_start();
        echo "Welcome :" . $_SESSION['username']."<br>";
        echo "Your Password is:".$_SESSION['password'];
?>
```

OUTPUT:

```
Welcome:abc

Your Password is:123
```

- Destroy a session:
  A php session is destroyed by session_destroy() fucntion. This function does not need any argument and a single call can destroy all session variables. You can also use unset() function to unset session variables.
  Example:
```
<?php
```

```
        unset($_SESSION['counter']);
?>
OR
<?php
        session_destroy();

?>
```

4.6. Cookies: setting a cookies, accessing cookies data and destroying cookies

- **Cookies** are text files stored on the **client** computer and they are kept of user tracking purpose. PHP transparently supports HTTP cookies.
- 3 steps involved in identifying returning users:
  - ➢ Server script sends a set of cookies to the browser. Ex: name, age
  - ➢ Browser stored 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.
- setcookie() function is used to set cookie. This function requires upto six arguments and called before <html> tag.

Syntax: setcookie(name,value,expire,path,domain,security);

Name: set the name of cookie and stored in environment variable called HTTP_COOKIE_VARS.

Value: set the value of named variable and content that you want to store.

Expiry: Specify a future time in seconds since 00:00:00 GMT on 1stjan 1970. If this parameter will not set then cookie will automatically expire.

Path: Specify directories for which cookie is valid.

Domain: Specify 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.

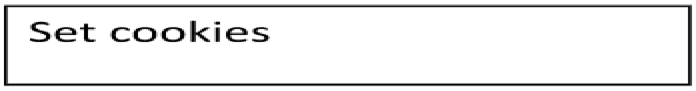
Security: if set to 1 then specify that cookie sent by secure transmission using HTTPS otherwise set to 0 means cookie sent by regular HTTP.

Example: create 2 cookies name and age and these coolies will expire after one hour.

Code: setcookie.php

```
<?php

        setcookie("name","abc", time()+3600);

        setcookie("age", "25", time()+3600, "/", "",0);

?>

<html><head><title></title></head>

<body>      <?php echo "set cookies"; ?></body></html>
```

OUTPUT


```
Set cookies
```
:

- **Accessing cookies with php:**

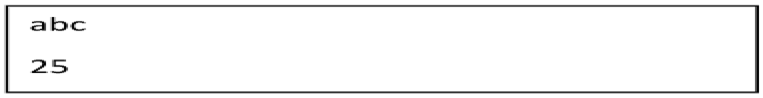
PHP provides many ways to access cookies. Simplest way is to use $_COOKIES or $HTTP_COOKIE_VARS variables.

Example:

Code: method1.php

```
<html><head><title>Accessing cookies with php</title></head>

<body>

        <?php echo $_COOKIE["name"] . "<br/>";

        echo $_COOKIE["age"] . "<br/>"; ?>

</body></html>
```

OUTPUT:


```
abc
25
```

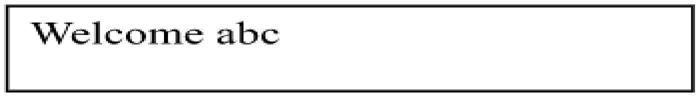
- **Check cookie is set or not with php:**

Example:

Code: cookie.php

```
<html><head><title></title></head>

<body>

        <?php  if(isset($_COOKIE["name"]))

                        echo "Welcome".$_COOKIE["name"] . "<br/>";

                else

                        echo "Sorry cookie not recognized  <br/>"; ?>

</body></html>
```

OUTPUT:


```
Welcome abc
```

- **Delete cookiewith php:**

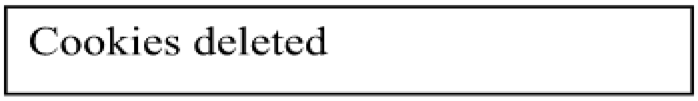
To delete a cookie call setcookie() function with name argument only this does not always work well. So it is safest to set the cookie with a date that has already expired:

Example:

Code: deletecookie.php

```
<?php
        setcookie(“name”,””,time()-60);
        setcookie(“age”,””,time()-60);
?>
<html><head><title>Deleteingcookites</title></head>
<body>
        <?php echo “Cookies deleted”; ?>
</body>
</html>
```

OUTPUT:



```
Cookies deleted
```