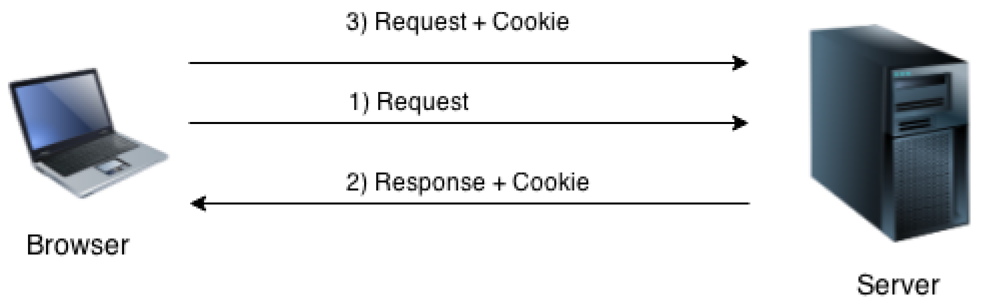
4. Cookies and Session

CO - ITG408-4: Perform PHP programs using cookies, sessions Marks 10

4.1 Cookies – Use of cookies, Attributes of cookies, create cookies, modify cookies value and delete cookies

A **cookie** in PHP is a small file with a maximum size of 4KB that the web server stores on the client computer. They are typically used to keep track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Setting Cookie In PHP: To set a cookie in PHP, the **setcookie()** function is used. The setcookie() function needs to be called prior to any output generated by the script otherwise the cookie will not be set.

Syntax:

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

Parameters: The setcookie() function requires six arguments in general which are:

- **Name:** It is used to set the name of the cookie.
- **Value:** It is used to set the value of the cookie.
- **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path:** It is used to specify the path on the server for which the cookie will be available.A single forward slash character permits the cookie to be valid for all directories.
- **Domain:** It is used to specify the domain for which the cookie is available.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:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.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.

Example

```
setcookie("name", "richa", time()+3600, "/","", 0);
```

Below are some operations that can be performed on Cookies in PHP:

- **Creating Cookies**: Creating a cookie named Auction_Item and assigning the value Luxury Car to it. The cookie will expire after 2 days(2 days * 24 hours * 60 mins * 60 seconds).

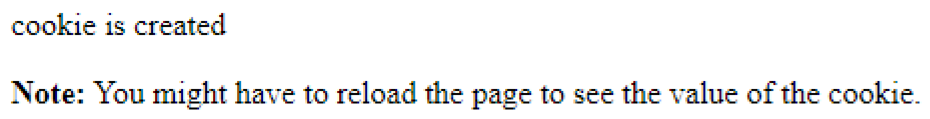
Example: This example describes the creation of the cookie in PHP.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24
* 60 * 60);
?>
<html>
<body>
    <?php
        echo "cookie is created."
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the
        page to see the value of the cookie.
    </p>

</body>
</html>
```

Note: Only the name argument in the setcookie() function is mandatory. To skip an argument, the argument can be replaced by an empty string(“”).

Output:



```
cookie is created

Note: You might have to reload the page to see the value of the cookie.
```

Cookie creation in PHP

Checking Whether a Cookie Is Set Or Not: It is always advisable to check whether a cookie is set or not before accessing its value. Therefore to check whether a cookie is set or not, the PHP isset() function is used. To check whether a cookie “Auction_Item” is set or not, the isset() function is executed as follows:

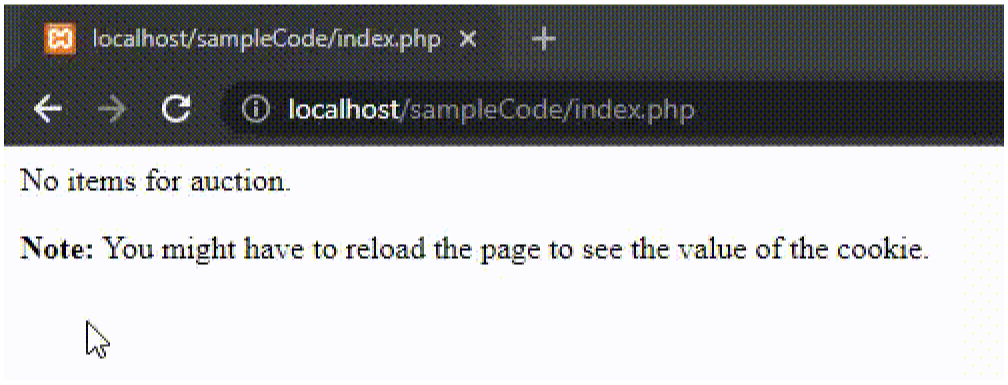
Example: This example describes checking whether the cookie is set or not.

-

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24
* 60 * 60);
?>
<html>
<body>
    <?php
    if (isset($_COOKIE["Auction_Item"]))
    {
        echo "Auction Item is a  " .
$_COOKIE["Auction_Item"];
    }
    else
    {
        echo "No items for auction.";
    }
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>

</body>
</html>
```

Output:


localhost/sampleCode/index.php

No items for auction.

Note: You might have to reload the page to see the value of the cookie.

Checking for the cookie to be set

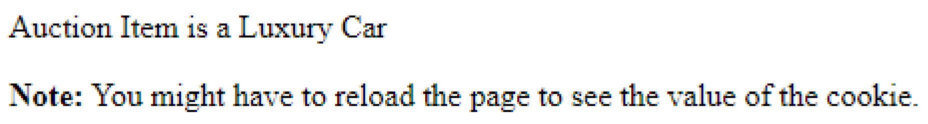
Accessing Cookie Values: For accessing a cookie value, the PHP $_COOKIE superglobal variable is used. It is an associative array that contains a record of all the cookies values sent by the browser in the current request. The records are stored as a list where the cookie name is used as the key. To access a cookie named “Auction_Item”, the following code can be executed.

Example: This example describes accessing & modifying the cookie value.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24
* 60 * 60);
?>
<html>
<body>
<?php
    echo "Auction Item is a  " . $_COOKIE["Auction_Item"];
?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>
```

```
</body>
</html>
```

Output:



```
Auction Item is a Luxury Car

Note: You might have to reload the page to see the value of the cookie.
```

Accessing the Cookie value

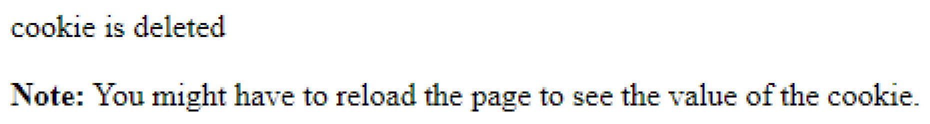
Deleting Cookies: The setcookie() function can be used to delete a cookie. For deleting a cookie, the setcookie() function is called by passing the cookie name and other arguments or empty strings but however this time, the expiration date is required to be set in the past. To delete a cookie named “Auction_Item”, the following code can be executed.

Example: This example describes the deletion of the cookie value.

```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24
* 60 * 60);
?>
<html>
<body>
    <?php
       setcookie("Auction_Item", "", time() - 60);
    ?>
    <?php
        echo "cookie is deleted"
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the page
        to see the value of the cookie.
    </p>
```

```
</body>
</html>
```

Output:



```
cookie is deleted

Note: You might have to reload the page to see the value of the cookie.
```

Deleting the Cookie

Important Points:

- If the expiration time of the cookie is set to 0 or omitted, the cookie will expire at the end of the session i.e. when the browser closes.
- The same path, domain, and other arguments should be passed that were used to create the cookie in order to ensure that the correct cookie is deleted.

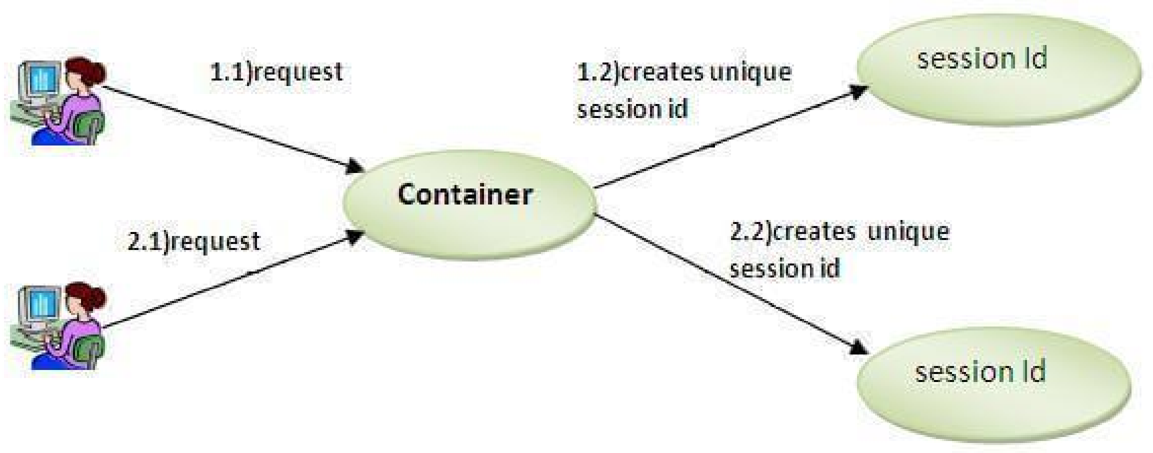
PHP Sessions

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the users computer.

What is a PHP Session?

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.
Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
So; Session variables hold information about one single user, and are available to all pages in one application.

PHP session is used to store and pass information from one page to another temporarily (until user close the website).
PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.
PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



Start a PHP Session

A session is started with the session_start() function.It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax

```
bool session_start ( void )
```

Example

```
session_start();
```

Session variables are set with the PHP **global variable: $_SESSION.**

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The `session_start()` function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php

session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
```

```
?>

</body>
</html>
```

How does it work? How does it know it's me?

Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
```

```
</body>
</html>
```

PHP Session Example

File: session1.php

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

File: session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()`:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>
</body>
</html>
```

4.3 Sending E-mail

mail() function is a useful function of PHP to send email using PHP script. You will require to set up some configurations to use this 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 );
```

Here is the description for each parameters.

Sr.No	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**

	headers **Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)**
	parameters **Optional. Specifies an additional parameter to the send mail program**

Steps: 1. Open the php.ini file and modify the file with the following configurations

```
SMTP=smtp.gmail.com
smtp_port=587
sendmail_from =sender@gmail.com
sendmail_path = "C:\xampp\sendmail\sendmail.exe -t"
mail.add_x_header=On
```

2. Open the sendmail.ini file from the location, C:\xampp\sendmail and modify the file with the following configurations.

```
smtp_server=smtp.gmail.com
smtp_port=587
smtp_ssl=tls
error_logfile=error.log
debug_logfile=debug.log
auth_username='username@gmail.com'
auth_password='password'
```

3. Restart the Apache server.

4. Login to the Gmail account that is used in the configuration of sendmail.ini file. Enable the 2 step verification option of this account for sending email using the Gmail account.Then set app password. Use that app password in the sendmail.ini file.

Syntax:

```
bool mail (string $to, string $subject, string $message
[, mixed $additional_headers [, string
$additional_parameters ]])
```

This function can take four arguments. The first argument takes the receiver's email address. The second argument takes the subject of the email. The third argument takes the email body. The last argument is optional and it contains additional information of the email as a string or an array.

Sending email using mail() function:

Different uses of mail() function are shown in this section by using multiple examples.

Example-1: Send a simple text email

How you can send a simple text email using mail() function is shown in this tutorial. Create a PHP file with the following script and run the script from any local or online server.

```php
<?php

//Email address of the receiver
$to = "user@gmail.com";
//Email subject
$subject = "testing mail() function";
//Email message
$message = "Sending email using mail() function";
//Header information
$headers = "From: Sender <xyz@gmail.com>\r\n";
//Send email
if(mail($to, $subject, $message, $headers))
   echo "Email sent successfully.";
else
   echo "Unable to send the email.";
?>
```