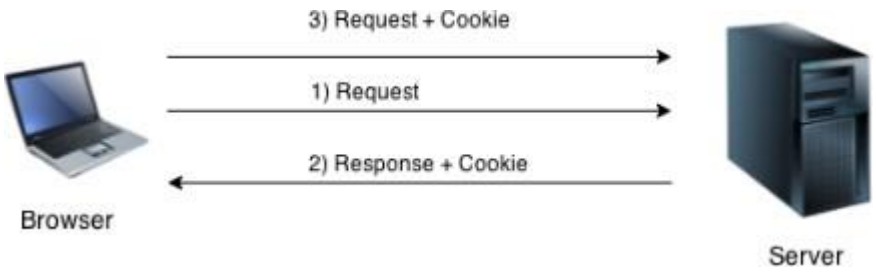
PHP Cookie

- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.
- 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.



- A cookie is a small text file that is saved on the user's computer. The maximum file size for a cookie is 4KB. It is also known as an HTTP cookie, a web cookie, or an internet cookie.
- When a user first visits a website, the site sends data packets to the user's computer in the form of a cookie.
- The information stored in cookies is not safe since it is kept on the client-side in a text format that anybody can see. We can activate or disable cookies based on our needs.

PHP Cookie (contd..)

PHP setcookie() function

- PHP setcookie() function is used to set cookie with HTTP response.
  Once cookie is set, you can access it by $_COOKIE superglobal variable.
- **Syntax**

bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path
[, string $domain [, bool $secure = false [, bool $httponly = false]]]]]])

- **Example**

```
setcookie("CookieName", "CookieValue");
/* defining name and value only*/
setcookie("CookieName", "CookieValue", time()+1*60*60);
//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)
setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "
mydomain.com", 1);
```

PHP Cookie (contd..)

PHP $_COOKIE

- PHP $_COOKIE superglobal variable is used to get cookie.
- **Example**

```
$value=$_COOKIE["CookieName"];//returns  cookie value
```

Example: Cookie

```
<?php
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
   echo "Sorry, cookie is not found!";
}
else
{
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

Sorry, cookie is not found!

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Cookie Value: Sonoo

PHP Cookie (contd..)

PHP Delete Cookie

- If you set the expiration date in past, cookie will be deleted.
- Example:

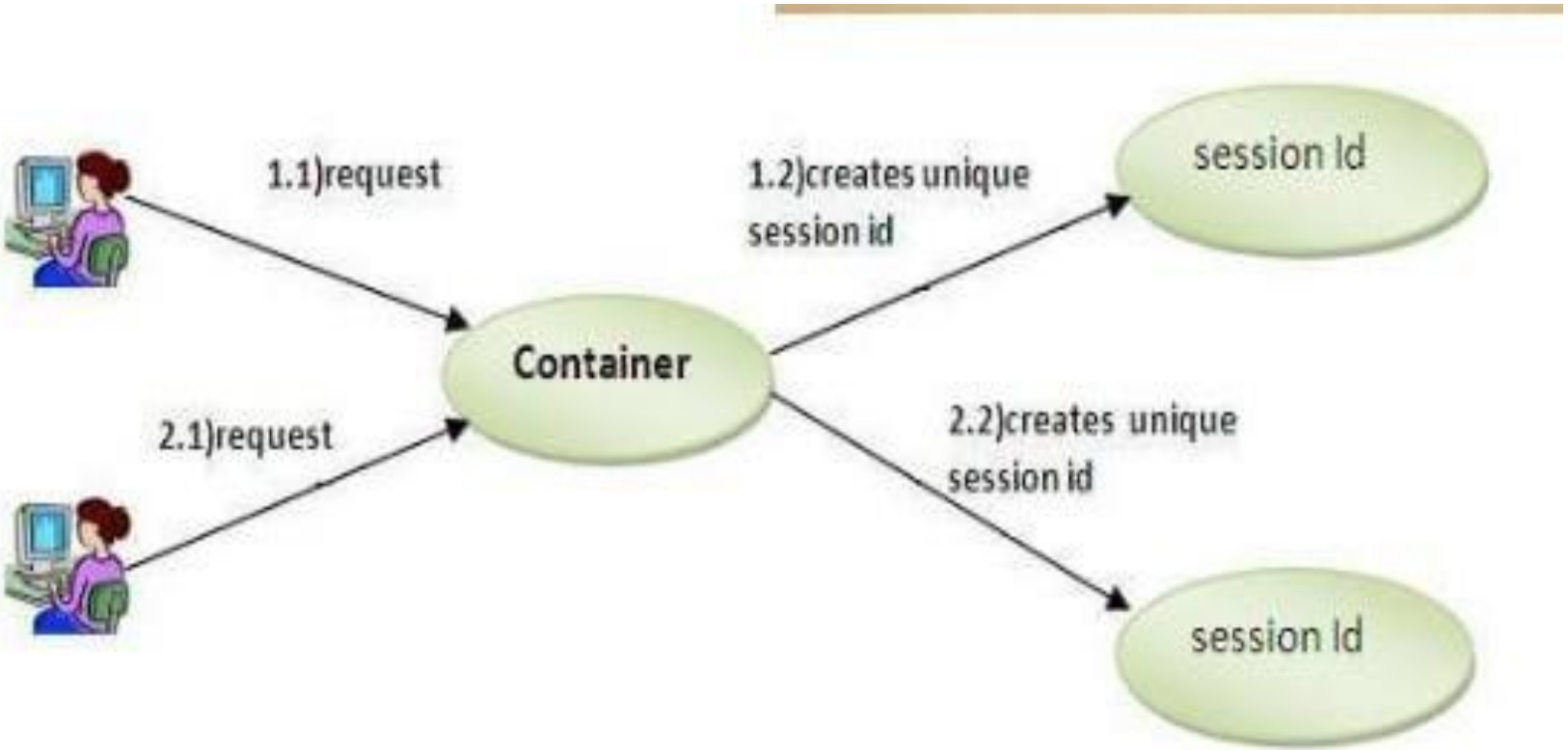
```
<?php
setcookie ("CookieName", "", time() -3600);
// set the expiration date to one hour ago
?>
```

PHP Session

- 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.

- A session is used to save information on the server momentarily so that it may be utilized across various pages of the website. It is the overall amount of time spent on an activity.
- The user session begins when the user logs in to a specific network application and ends when the user logs out of the program or shuts down the machine.
- Session values are far more secure since they are saved in binary or encrypted form and can only be decoded at the server. When the user shuts down the machine or logs out of the program, the session values are automatically deleted.
- We must save the values in the database to keep them forever.

PHP Session


1.1)request
1.2)creates unique session id
session Id
Container
2.1)request
2.2)creates unique session id
session Id

PHP Session

PHP session_start() function

- PHP session_start() function is used to start the session. 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();
```

PHP Session

PHP $_SESSION

- PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.
- **Example: Store information**

```
$_SESSION["user"] = "Sachin";
```

- **Example: Get information**

```
echo $_SESSION["user"];
```

Example: PHP Session

Session1.php

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

</body>
</html>
```

Session2.php

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

PHP Session Counter Example

```
<?php  session_start();

  if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
  echo ("Page Views: ".$_SESSION['counter']);
?>
```

PHP Session

PHP Destroying Session

- PHP session_destroy() function is used to destroy all session variables completely.

```
<?php
session_start();
session_destroy();
?>
```

Difference Between Session and Cookies

Cookie	Session
Cookies are client-side files on a local computer that hold user information.	Sessions are server-side files that contain user data.
Cookies end on the lifetime set by the user.	When the user quits the browser or logs out of the programmed, the session is over.
It can only store a certain amount of info.	It can hold an indefinite quantity of data.
The browser's cookies have a maximum capacity of 4 KB.	We can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time.
Because cookies are kept on the local computer, we don't need to run a function to start them.	To begin the session, we must use the session start() method.

Exercise

- **Part1: Keep me logged in!**
  Starting with the login form , add the following: A successful login should set some session variable so that the server knows that the user is logged in. For example, set $_SESSION['loggedin'] to be TRUE.
- When the page is loaded, check the session variable. If the user is logged in, display the welcome message instead of the login form.
- Add a "Log Out" button to the welcome message that, when clicked, removes the session variable so that the user is logged out.
- Clicking the button should redirect the user to the same page, which now shows a login form. This behaviour should be consistent with the lifetime of sessions. That is, the user should stay logged in despite refreshing and opening the same page in multiple tabs, but should be logged out once the browser is closed.

- **Part 2: Remember me!**
  Add a "Remember me!" checkbox to the login form. If the box is checked and the login is successful, save a cookie that identifies the user to the server. On further visits to the page, the user should appear logged in, even if the browser has been closed.
- You may choose a reasonable expiration time for the cookie. Remember also that if the user manually logs out by clicking the "Log Out" button that the cookie should be deleted (set the expiration to be some time in the past).