Cookies, Sessions, and Authentication. Introduction to XML, Ajax, JQuery, Browsers and the DOM

Working with Session

In this section, we are going to see the following to work with PHP sessions. These are,

- Starting session
- Storing session variable
- Accessing session variable
- Clearing session variable

Starting Session

PHP has a built-in function named as *session_start()*. This function is used to start a new session or to resume an existing session. By starting the session, it will call session handlers to read and return current session data by using the callback functions. As of PHP version 7, this function can have an associative array of options as its argument. Using these options, we can overwrite the default session configurations.

Storing Session Variable

The $_SESSION array is used to create a session variable. In the following code, I have created a new session with the index named as *tagName*.

```
session_start();

$_SESSION["tagName"] = "PHP";
```

Accessing Session Variable

Once the data is stored in a session variable, then it is in global scope and can be accessed from any PHP file.

```
session_start();

$tag = $_SESSION["tagName"];

echo "Welcome to $tag world!";
```

Clearing Session Variable

PHP provides functions to clear existing session data. Those are, *session_destroy()*, *session_unset()* and more.

The *session_destroy()* function is used to clear all the current session data. After destroying the session, we have to reload the page to see that the session is cleared.

To make the session_destroy() action to show immediate effect, we can call session_unset() or unset() function to clear session data. Code shows an example to clear session variable.

```
session_start();

session_destroy();

echo $_SESSION["tagName"];//session remains until refresh

unset($_SESSION["tagName"]);

echo $_SESSION["tagName"];//session no more;
```

PHP Cookies

In this section, we are going to learn how to set data to the cookies array and use them all over the site.

Setting cookies

In PHP, we can set cookies by using setcookie*(), setrawcookie() or header()* function. setrawcookie*()* is same as setcookie*()*, but differs by setting the raw value of the cookie with the header without encoding. These function must be used before sending any output to the browser. Otherwise, the cookie data will not be set with the header information.

While setting cookies, it needs information like cookies name, domain, expiration time and more. The following Code shows the usage example for setting cookies by using the header() and setcookie() functions.

```
header("Set-Cookie: platform=php; expires=Mon, 20-April-13 17:30:48 GMT; path=/; domain=phppot.com");
```

or

```
setcookie("platform", "php", time()+7200, "/", ".phppot.com", 0);
```

Accessing cookies

PHP cookies can be accessed by using $_COOKIE variable. Also, we can use the other superglobals like $_SERVER, $_ENV and PHP getenv() function to access cookies by specifying HTTP_COOKIE index as shown in the code below. But, $_COOKIE has the guaranteed access in all the server compare to the other global variables. Because some server configuration will not allow us access $_SERVER, $_ENV variables.

```
echo $_COOKIE["platform"]; // Output: php

// OR

echo $_SERVER[‘HTTP_COOKIE’]; // Output: platform=php

echo getenv(‘HTTP_COOKIE’); // Output: platform=php
```

Introduction to XML

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything.

This note is a note to Tove from Jani, stored as XML:

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The XML above is quite self-descriptive:

- It has sender information.
- It has receiver information
- It has a heading
- It has a message body.

But still, the XML above does not DO anything. XML is just information wrapped in tags. Someone must write a piece of software to send, receive, store, or display it:

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

What is jQuery

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

Will jQuery work in all browsers?

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers.

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

`$(this).hide()` - hides the current element.

`$("p").hide()` - hides all <p> elements.

`$(".test").hide()` - hides all elements with class="test".

`$("#test").hide()` - hides the element with id="test".