USN | | | | | | | | | |

Internal Assessment Test 3 – Jan. 2022

Sub:	Web Technologies & its Applications					Sub Code:	17CS71
Date:	24-01-2022	Duration:	90 min's	Max Marks:	50	Sem / Sec:	7 – D

Answer any FIVE FULL Questions

		MARKS	CO	
1(a)	Briefly explain Arrays in PHP with examples.	[05]	CO3	L2

PHP Arrays

PHP supports arrays. An arrayis a data structure that allows the programmer to collect a number of related elementstogether in a single variable.an array is actually an **ordered map**, which associates each value in the arraywith a key.

$days	0	1	2	3	4
Keys					
	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"
Values					

FiGURE 9.1 Visualization of a key-value array

Array keys in most programming languages are limited to integers, start at 0, and go up by 1. In PHP, keys *must* be either integers or strings and need not be sequential. This means you cannot use an array or object as a key.

One should be especially careful about mixing the types of the keys for an array since PHP performs cast operations on the keys that are not integers or strings. You cannot have key "1" distinct from key 1 or 1.5, since all three will be cast to the integer key 1.

Array values, unlike keys, are not restricted to integers and strings. They can be

any object, type, or primitive supported in PHP. You can even have objects of your

own types, so long as the keys in the array are integers and strings.

Defining and Accessing an Array

The following declares an empty array named days:

```
$days = array();
```

To define the contents of an array as strings for the days of the week as shown

in Figure 9.1, you declare it with a comma-delimited list of values inside the ( ) braces using either of two following syntax's:

```
$days = array("Mon","Tue","Wed","Thu","Fri");
$days = ["Mon","Tue","Wed","Thu","Fri"]; // alternate syntax
```

no keys are explicitly defined for the array, thedefault key values are 0, 1, 2, . . . , n.echoes the value of our $days array for the key=1, which results inoutput of Tue.

```
echo "Value at index 1 is ". $days[1]; // index starts at zero
```

You could also define the array elements individually using this same squarebracket notation:

```
$days = array();
$days[0] = "Mon";
$days[1] = "Tue";
$days[2] = "Wed";
// also alternate approach
$daysB = array();
$daysB[] = "Mon";
$daysB[] = "Tue";
$daysB[] = "Wed";
```

```
$days = array(0 =>"Mon", 1 =>"Tue", 2 =>"Wed", 3 =>"Thu", 4=>"Fri");
0->key
Mon->value
```

Figure 9.2 Explicitly assigning keys to array elements

```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
Key->mon
Value->40
echo $forecast["Tue"]; // outputs 47
echo $forecast["Thu"]; // outputs 40
```

consider an array to be a dictionary or hash map. These types of arraysin PHP are generally referred to as **associative arrays**.

to access an element in an associative array, yousimply use the key value rather than an index:

```
echo $forecast["Wed"]; // this will output 52
```

Multidimensional Arrays

PHP also supports multidimensional arrays.

```
$month = array
(
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri")
);
echo $month[0][3]; // outputs Thu
$cart = array();
$cart[] = array("id" => 37, "title" =>"Burial at Ornans",
"quantity" => 1);
$cart[] = array("id" => 345, "title" =>"The Death of Marat",
"quantity" => 1);
$cart[] = array("id" => 63, "title" =>"Starry Night", "quantity" => 1);
echo $cart[2]["title"]; // outputs Starry Night
```

listing 9.1 Multidimensional arrays

Iterating through an Array

One of the most common programming tasks that you will perform with an array

is to iterate through its contents.

```
// while loop
```

```
$i=0;
while ($i < count($days)) {
echo $days[$i] . "<br>";
$i++;
}
// do while loop
$i=0;
do {
echo $days[$i] . "<br>";
$i++;
} while ($i < count($days));
// for loop
for ($i=0; $i<count($days); $i++) {
echo $days[$i] . "<br>";
}
```

Code: Iterating through an array using while, do while, and for loops

output thecontent of the $days array using the built-in function count()

For an associative array,the foreach loop and illustrated for the $forecast array

```
// foreach: iterating through the values
foreach ($forecast as $value) {
echo $value . "<br>";
}
// foreach: iterating through the values AND the keys
foreach ($forecast as $key => $value) {
echo "day" . $key . "=" . $value;
}
```

Adding and Deleting Elements

In PHP, arrays are dynamic, that is, they can grow or shrink in size. An element can be added to an array simply by using a key/index that hasn’t been used, as shown below:

$days[5] = "Sat";

Since there is no current value for key 5, the array grows by one, with the new

key/value pair added to the end of our array. If the key had a value already, the same

style of assignment replaces the value at that key. As an alternative to specifying the

index, a new element can be added to the end of any array using the following

technique:

$days[ ] = "Sun";

$days = array("Mon","Tue","Wed","Thu","Fri");

$days[7] = "Sat";

print_r($days);

What will be the output of the print_r()? It will show that our array now

contains the following:

Array ([0] => Mon [1] => Tue [2] => Wed [3] => Thu [4] => Fri [7] => Sat)

Referencing$days[6], for instance, it will return a NULL value, which is a special PHP valuethat represents a variable with no value.You can also create “gaps” by explicitly deleting array elements using theunset() function,

$days = array("Mon","Tue","Wed","Thu","Fri");

unset($days[2]);

unset($days[3]);

print_r($days); // outputs: Array ([0] => Mon [1] => Tue [4] => Fri)

$days = array_values($days);

print_r($days); // outputs: Array ([0] => Mon [1] => Tue [2] => Fri)

you can remove “gaps” in arrays (whichreally are just gaps in the index keys) using the array_values() function, whichreindexes the array numerically.

Checking If a Value Exists

To checkif a value exists for a key, you can therefore use the isset() function, which returns

true if a value has been set, and false otherwise.

```
$oddKeys = array (1 =>"hello", 3 =>"world", 5 =>"!");
if (isset($oddKeys[0])) {
// The code below will never be reached since $oddKeys[0] is not set!
echo "there is something set for key 0";
}
if (isset($oddKeys[1])) {
// This code will run since a key/value pair was defined for key 1
echo "there is something set for key 1, namely ". $oddKeys[1];
}
```

1(b) Explain the different ways of error handling in PHP with examples. [05] CO3 L2

Procedural Error Handling

In the procedural approach to error handling, the programmer needs to explicitly test for error conditions after performing a task that might generate an error.

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);
$error = mysqli_connect_error();
if ($error != null) {
// handle the error
...
}
```

Object-Oriented Exception Handling

When a runtime error occurs, PHP *throws* an *exception*. This exception can be *caught* and handled either by the function, class, or page that generated the exception or by the code that called the function or class. If an exception is not caught, then eventually the PHP environment will handle it by terminating execution with an “Uncaught Exception” message.4

```
// Exception throwing function
function throwException($message = null,$code = null) {
throw new Exception($message,$code);
}
try {
// PHP code here
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS,
DBNAME)
or throwException("error");
//...
}
catch (Exception $e) {
echo ' Caught exception: ' . $e->getMessage();
echo ' On Line : ' . $e->getLine();
echo ' Stack Trace: '; print_r($e->getTrace());
} finally {
// PHP code here that will be executed after try or after catch
}
```

The finally block is optional. Any code within it will always be executed *after* the code in the try or in the catch blocks, even if that code contains a return statement. It is typically used if the developer wants certain things done regardless of whether an exception occurred, such as closing a connection or removing temporary files.

```
function processArray($array)
{
// make sure the passed parameter is an array with values
if ( empty($array) ) {
throw new Exception('Array with values expected');
}
// process the array code
```

```
...
}
```

One possible strategy for such a scenario is to throw an exception:

```
public function setBirthDate($birthdate){
    // set variable only if passed a valid date string
    if ( $timestamp = strtotime($birthdate) ) {
    $this->birthDate=$timestamp;
    }
    else {
    throw new Exception("Invalid Date in Artist->setBirthDate()");
    }
    }


    try {
    // PHP code here
    }
    catch     (Exception $e) {
    // do some application-specific exception handling here
    ...
    // now rethrow exception
    throw $e;
    }
```

2(a)	Explain selectors in JQuery with examples. **jQuery Selectors** **Basic Selectors** The four basic selectors were defined, and include the universalselector, class selectors, id selectors, and elements selectors. To review: ▪ $("*") **Universal selector** matches all elements (and is slow). ▪ $("tag") **Element selector** matches all elements with the given element name.	[10]	CO5	L2

- $(".class") **Class selector** matches all elements with the given CSS class.
- $("#id") **Id selector** matches all elements with a given HTML id attribute.

For example, to select the single element with id="grab"you wouldwrite:

```
var singleElement = $("#grab");
```

To get a set of all the <a>elements the selector would be:

```
var allAs = $("a");
```

These selectors are powerful enough that they can replace the use ofgetElementById() entirely.

Attribute Selector

An **attribute selector** provides a way to select elements by either the presence of an element attribute or by the value of an attribute.

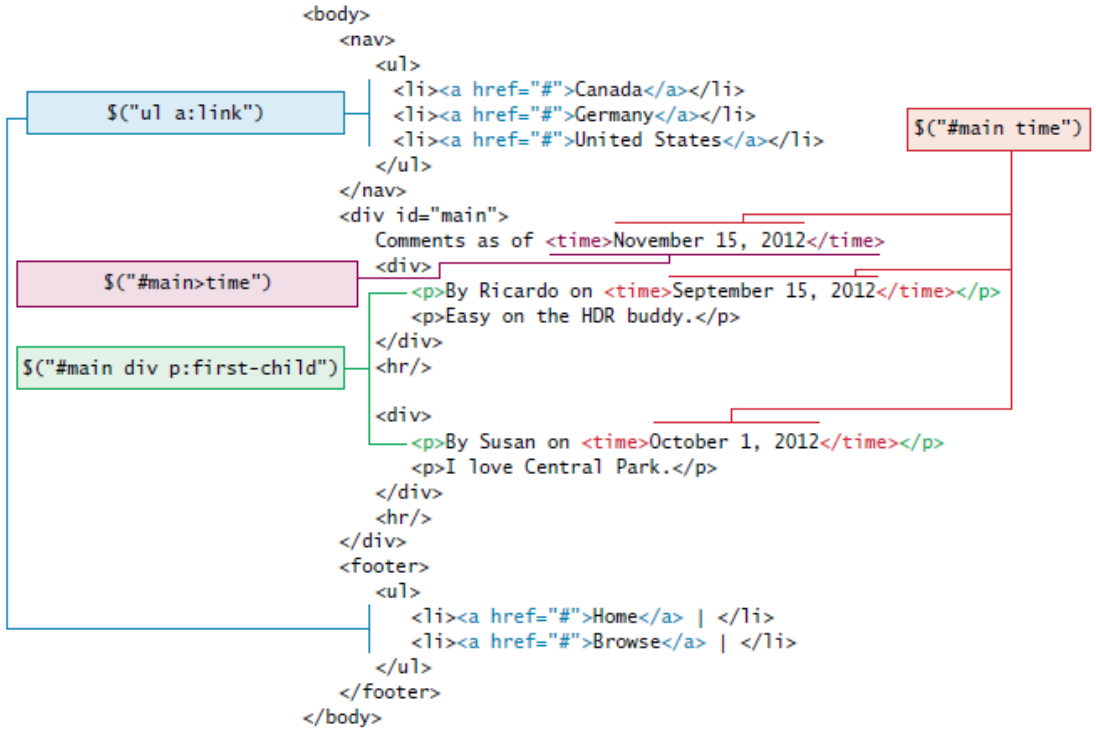


FIGURE 15.4 Illustration of some jQuery selectors and the HTML being selected

A list of sample CSS attribute selectorswas given in Chapter 3 (Table 3.4), but to jog your memory with an example, considera selector to grab all <img>elements with an src attribute beginning with /artist/ as:

```
var artistImages = $("img[src^='/artist/']");
```

Recall that you can select by attribute with square brackets ([attribute]), specifya value with an equals sign ([attribute=value]) and search for a particular value inthe beginning, end, or anywhere inside a string with ^, $, and * symbols respectively.

([attribute^=value], [attribute$=value], [attribute*=value]).

Pseudo-Element Selector

Pseudo-elements are special elements, which are special cases of regular ones. these **pseudo-element selectors** allow you to appendto any selector using the colon and one of :link, :visited, :focus, :hover,:active, :checked, :first-child, :first-line, and :first-letter.

These selectors can be used in combination with the selectors presented above,or alone. Selecting all links that have been visited, for example, would be specified

with:

```
var visitedLinks = $("a:visited");
```

Contextual Selector

Another powerful CSS selector included in jQuery's selection mechanism is the**contextual selectors** introduced in Chapter 3. These selectors allowed you to specifyelements with certain relationships to one another in your CSS. These relationshipsincluded descendant (space), child (>), adjacent sibling (+), and general sibling (~).To select all <p>elements inside of elements you would write

```
var para = $("div p");
```

Content Filters

The **content filter** is the only jQuery selector that allows you to append filters to all ofthe selectors you've used thus far and match a particular pattern. You can selectelements that have a particular child using :has(), have no children using :empty, ormatch a particular piece of text with :contains(). Consider the following example:

```
var allWarningText = $("body *:contains('warning')");
```


```
var allWarningText = $("body *:contains('warning')");
```


It will return a list of all the DOM elements with the word *warning* inside of

them. You might imagine how we may want to highlight those DOM elements by

coloring the background red as shown in Figure 15.5 with one line of code:

```
$("body *:contains('warning')").css("background-color", "#aa0000");
```

Selector	CSS Equivalent	Description
$(:button)	$("button, input[type='button']")	Selects all *buttons*.
$(:checkbox)	$('[type=checkbox]')	Selects all *checkboxes*.
$(:checked)	No equivalent	Selects elements that are checked. This includes radio buttons and checkboxes.
$(:disabled)	No equivalent	Selects form elements that are disabled. These could include <button>, <input>, <optgroup>, <option>, <select>, and <textarea>
$(:enabled)	No equivalent	Opposite of :disabled. It returns all elements where the disabled attribute=false as well as form elements with no disabled attribute.
$(:file)	$('[type=file]')	Selects all elements of type file.
$(:focus)	$(document.activeElement)	The element with focus.
$(:image)	$('[type=image]')	Selects all elements of type image.
$(:input)	No equivalent	Selects all <input>, <textarea>, <select>, and <button> elements.
$(:password)	$('[type=password]')	Selects all password fields.
$(:radio)	$('[type=radio]')	Selects all radio elements.
$(:reset)	$('[type=reset]')	Selects all the reset buttons.
$(:selected)	No equivalent	Selects all the elements that are currently selected of type <option>. It does not include checkboxes or radio buttons.
$(:submit)	$('[type=submit]')	Selects all submit input elements.
$(:text)	No equivalent	Selects all input elements of type text. $('[type=text]') is almost the same, except that $(:text) includes <input>. fields with no type specified.

TABLE 15.1 jQuery form selectors and their CSS equivalents when applicable

Form Selectors

Since form HTML elements are well known and frequently used to collect andtransmit data, there are jQuery selectors written especially for them. These selectors,listed in Table 15.1, allow for quick access to certain types of field as well as fieldsin certain states.attributes like the href attribute of an <a>tag, the src attribute of an <img>, or theclass attribute of most elements.In jQuery we can both set and get an attribute value by using the attr() methodon any element from a selector. This function takes a parameter to specify whichattribute, and the optional second parameter is the value to set it to. If no secondparameter is passed, then the return value of the call is the current value of theattribute. Some example usages are:

```
// var link is assigned the href attribute of the first <a> tag
var link = $("a").attr("href");
// change all links in the page to http://funwebdev.com
$("a").attr("href","http://funwebdev.com");
// change the class for all images on the page to fancy
$("img").attr("class","fancy");
```

3(a)	Explain super global array $_FILES with examples. The $_FILES associative array contains items that have been uploaded to the current script. The element is used to create the user interface for uploading a file from the client to the server. The user interface is only one part of the uploading process. A server script must process the upload file(s) in some way; the $_FILES array helps in this process. HTML Required for File Uploads To allow users to upload files, there are some specific things you must do: ■ First, you must ensure that the HTML form uses the HTTP POST method, since transmitting a file through the URL is not possible. ■ Second, you must add the enctype="multipart/form-data" attribute to the HTML form that is performing the upload so that the HTTP request can submit multiple pieces of data (namely, the HTTP post body, and the HTTP file attachment itself). ■ Finally you must include an input type of file in your form. This will show up with a browse button beside it so the user can select a file from their computer to be uploaded. Submit Handling the File Upload in PHP The corresponding PHP file responsible for handling the upload will utilize the superglobal $_FILES array. This array will contain a key=value pair for each file uploaded in the post. The key for each element will be the name attribute from the HTML form, while the value will be an array containing information about the file as well as the file itself. The keys in that array are the name, type, tmp_name, error, and size. The values for each of the keys are described below. ■ name is a string containing the full file name used on the client machine, including any file extension. It does not include the file path on the client's machine. ■ type defines the MIME type of the file. This value is provided by the client browser and is therefore not a reliable field. ■ tmp_name is the full path to the location on your server where the file is being temporarily stored. The file will cease to exist upon termination of the script, so it should be copied to another location if storage is required. ■ error is an integer that encodes many possible errors and is set to UPLOAD_ERR_OK (integer value 0) if the file was uploaded successfully. ■ size is an integer representing the size in bytes of the uploaded file	[05]	CO4	L2
3(b)	Explain super global array $_GET and $_POST with a neat diagram. The $_GET and $_POST arrays are the most important superglobal variables in PHP since they allow the programmer to access data sent by the client in a query string.	[05]	CO4	L2

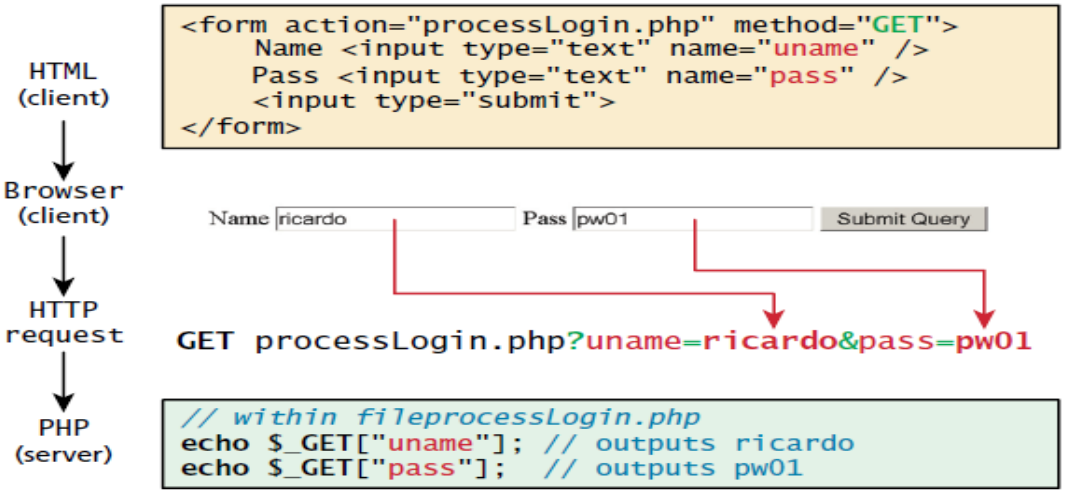


FIGURE 9.5 Illustration of flow from HTML, to request, to PHP's $_GET array

An HTML form (or an HTML link) allows a client to send data to the server. That data is formatted such that each value is associated with a name defined in the form. If the form was submitted using an HTTP GET request, then the resulting URL will contain the data in the query string. PHP will populate the superglobal $_GET array using the contents of this query string in the URL.

If the form was sent using HTTP POST, then the values would not be visible in the URL, but will be sent through HTTP POST request body. From the PHP programmer's perspective, almost nothing changes from a GET data post except that those values and keys are now stored in the $_POST array.

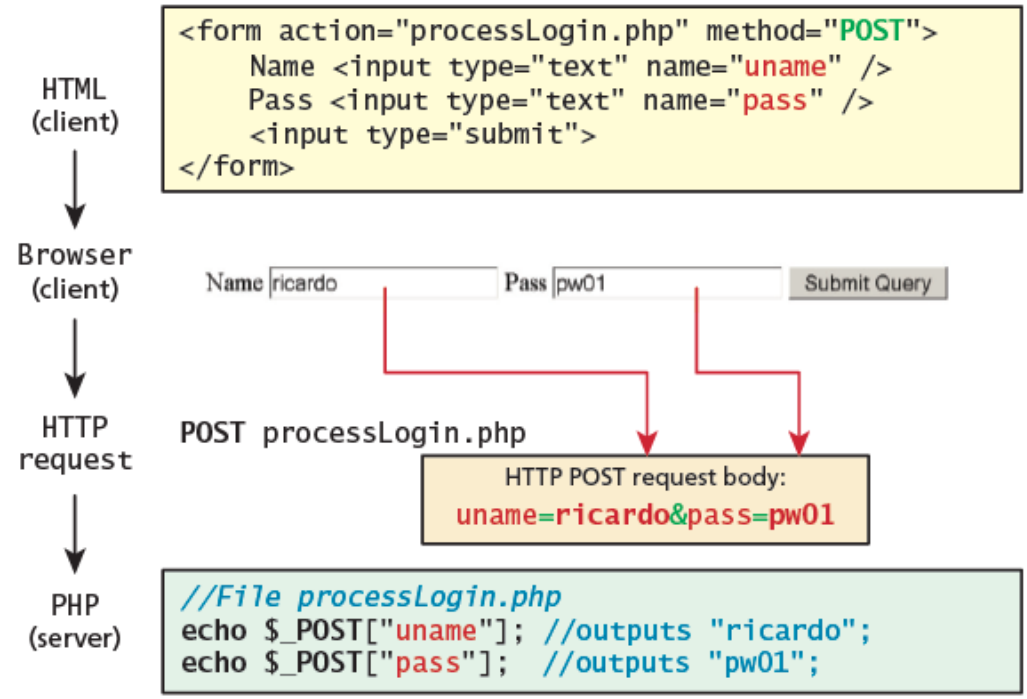


FIGURE 9.6 Data flow from HTML form through HTTP request to PHP's $_POST array

4(a) How cookies and session work? Give examples. [10] CO2 L2

While cookie information is stored and retrieved by the browser, the information ina cookie travels within the HTTP header. Figure 13.6 illustrates how cookies work.There are limitations to the amount of information that can be stored in acookie (around 4K) and to the number of cookies for a domain (for instance,Internet Explorer 6 limited a domain to 20 cookies).HTTP cookies can also expire. That is, the browser will delete cookies that

are beyond their expiry date (which is a configurable property of a cookie). If acookie does not have an expiry date specified, the browser will delete it when thebrowser closes (or the next time it accesses the site). For this reason, some commentatorswill say that there are two types of cookiessession cookies and persistentcookies. A **session cookie** has no expiry stated and thus will be deleted at the end ofthe user browsing session. **Persistent cookies** have an expiry date specified; they willpersist in the browser's cookie file until the expiry date occurs, after which they aredeleted.

The most important limitation of cookies is that the browser may be configuredto refuse them. As a consequence, sites that use cookies should not depend on theiravailability for critical features. Similarly, the user can also delete cookies or eventamper with the cookies, which may lead to some serious problems if not handled.

Several years ago, there was an instructive case of a website selling stereos and televisionsthat used a cookie-based shopping cart. The site placed not only the productidentifier but also the product price in the cart. Unfortunately, the site then used theprice in the cookie in the checkout. Several curious shoppers edited the price in thecookie stored on their computers, and then purchased some big-screen televisionsfor only a few cents!

Using Cookies

Like any other web development technology, PHP provides mechanisms for writingand reading cookies. Cookies in PHP are *created* using the setcookie() functionand are *retrieved* using the $_COOKIES superglobal associative array.Below example illustrates the writing of a persistent cookie in PHP

```
<?php
// add 1 day to the current time for expiry time
$expiryTime = time()+60*60*24;
// create a persistent cookie
$name = "Username";
$value = "Ricardo";
setcookie($name, $value, $expiryTime);
?>
```

The setcookie() function also supports several more parameters, which furthercustomize the new cookie. You can examine the online official PHP documentationfor more information. The below example illustrates the reading of cookie values. Notice that when we reada cookie, we must also check to ensure that the cookie exists. In PHP, if the cookiehas expired (or never existed in the first place), then the client's browser would not

send anything, and so the $_COOKIE array would be blank.

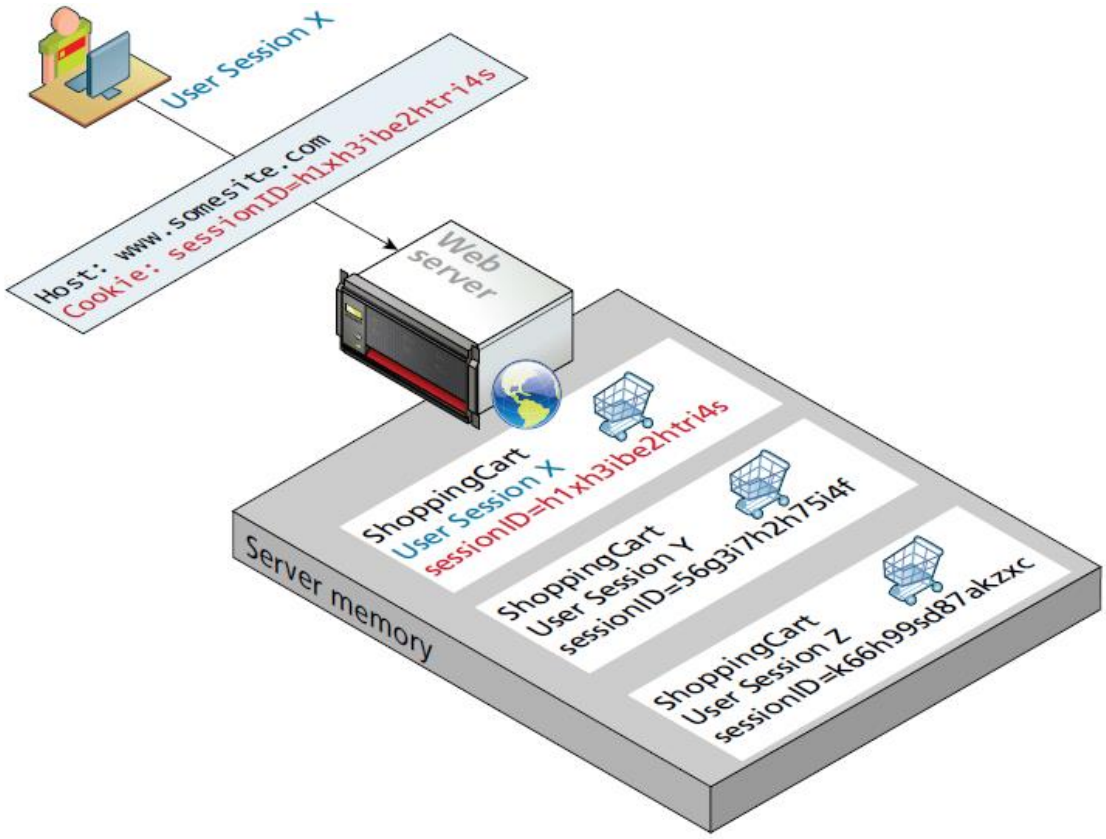
```
<?php
```

```
if( !isset($_COOKIE['Username']) ) {
//no valid cookie found
}
else {
echo "The username retrieved from the cookie is:";
echo $_COOKIE['Username'];
}
?>
```

How Does Session State Work?

The first thing to know about session state is that it works within the sameHTTP context as any web request. The server needs to be able to identify a givenHTTP request with a specific user request. Since HTTP is stateless, some type ofuser/session identification system is needed.

In PHP, this is a unique 32-byte string that is bydefault transmitted back and forth between the user and the server via a sessioncookie (see Section 13.4.1 above), as shown in Figure 13.9.



For a brand new session, PHP assigns an initially emptydictionary-style collection that can be used to hold any state values for this session.When the request processing is finished, the session state is saved to some type of

state storage mechanism, called a session state provider (discussed in next section).Finally, when a new request is received for an already existing session, the session'sdictionary collection is filled with the previously saved session data from the sessionstate provider.

5 (a)	How do you achieve encapsulation in PHP? Give examples. Data Encapsulation Perhaps the most important advantage to object-oriented design is the possibility of encapsulation, which generally refers to restricting access to an object's internal components. Another way of understanding encapsulation is: it is the hiding of an object's implementation details. A properly encapsulated class will define an interface to the world in the form of its public methods, and leave its data, that is, its properties, hidden (that is, private). This allows the class to control exactly how its data will be used. If a properly encapsulated class makes its properties private, then how do you access them? The typical approach is to write methods for accessing and modifying properties rather than allowing them to be accessed directly. These methods are commonly called getters and setters (or accessors and mutators). Some development environments can even generate getters and setters automatically. A getter to return a variable's value is often very straightforward and should not modify the property. It is normally called without parameters, and returns the property from within the class. For instance: public function getFirstName() { return $this->firstName; } Setter methods modify properties, and allow extra logic to be added to prevent properties from being set to strange values. For example, we might only set a date property if the setter was passed an acceptable date: The below example demonstrates how the Artist class could be used and tested	[05]	CO 4	L 2
5 (b)	Explain serialization with examples. **Serialization** is the process of taking a complicated object and reducing it down to zeros and ones for either storage or transmission. Later that sequence of zeros and ones can be reconstituted into the original object.	[05]	CO 5	L 2

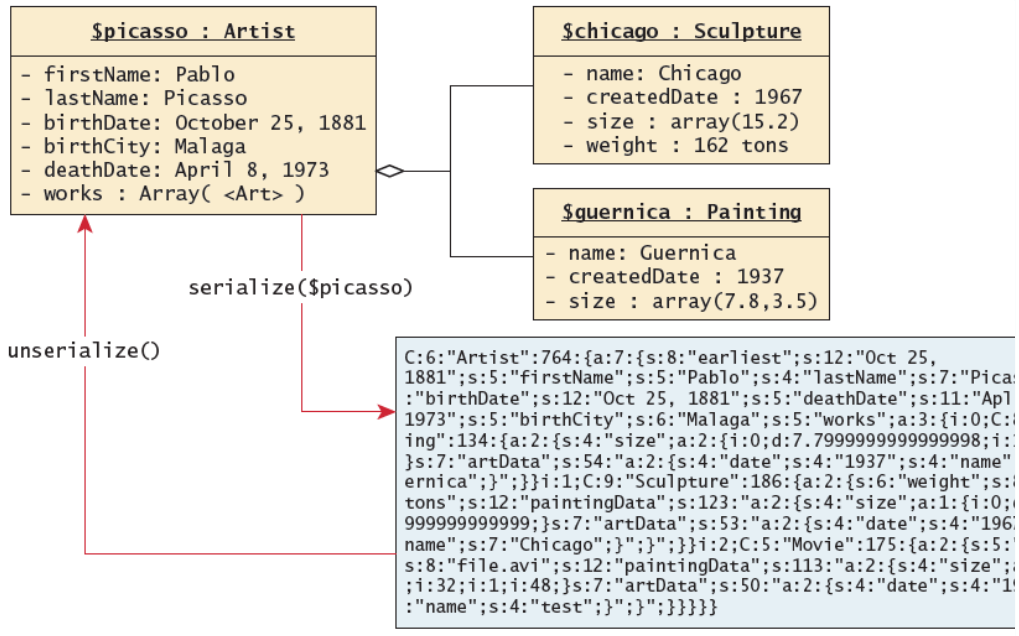


FIGURE 13.7 Serialization and deserialization

In PHP objects can easily be reduced down to a binary string using the serialize()function. The resulting string is a binary representation of the object and thereforemay contain unprintable characters. The string can be reconstituted back into anobject using the unserialize() method.While arrays, strings, and other primitive types will be serializable by default, classesof our own creation must implement the Serializable interface shown below.

which requires adding implementations for serialize() and unserialize() to anyclass that implements this interface.

```
interface Serializable {
        /* Methods */
        public function serialize();
        public function unserialize($serialized);
        }
```

shows how the Artist class must be modified to implement theSerializable interface by adding the implements keyword to the class definitionand adding implementations for the two methods.

listing 13.4 Artist class modified to implement the Serializable interface

```
class Artist implements Serializable {
//...
        // Implement the Serializable interface methods
        public function serialize() {
        // use the built-in PHP serialize function
        return serialize(
        array("earliest" =>self::$earliestDate,
        "first" => $this->firstName,
        "last" => $this->lastName,
        "bdate" => $this->birthDate,
        "ddate" => $this->deathDate,
        "bcity" => $this->birthCity,
        "works" => $this->artworks
        );
        );
        }
        public function unserialize($data) {
        // use the built-in PHP unserialize function
        $data = unserialize($data);
        self::$earliestDate = $data['earliest'];
        $this->firstName = $data['first'];
        $this->lastName = $data['last'];
        $this->birthDate = $data['bdate'];
        $this->deathDate = $data['ddate'];
        $this->birthCity = $data['bcity'];
        $this->artworks = $data['works'];
```

```
}
//...
}
```

If the data above is assignedto $data, then the following line will instantiate a new object identical to the

original:

```
$picassoClone = unserialize($data);
```

6 (a)	Briefly explain inheritance in PHP with neat class diagram. Inheritance Along with encapsulation, inheritance is one of the three key concepts in object oriented design and programming Inheritance enables you to create new PHP classes that reuse, extend, and modify the behavior that is defined in another PHP class. Although some languages allow it, PHP only allows you to inherit from one class at a time. A class that is inheriting from another class is said to be a subclass or a derived class. The class that is being inherited from is typically called a superclass or a base class. When a class inherits from another class, it inherits all of its public and protected methods and properties. Figure 10.9 illustrates how inheritance is shown in a UML class diagram. Just as in Java, a PHP class is defined as a subclass by using the extends keyword. class Painting extends Art { . . . } Referencing Base Class Members As mentioned above, a subclass inherits the public and protected members of the base class. Thus in the following code based on Figure 10.9, both of the references will work because it is as if the base class public members are defined within the subclass. $p = new Painting(); . . . // these references are ok echo $p->getName(); // defined in base class echo $p->getMedium(); // defined in subclass Unlike in languages like Java or C#, in PHP any reference to a member in the base class requires the addition of the parent:: prefix instead of the $this-> prefix. So within the Painting class, a reference to the getName() method would be: parent::getName() It is important to note that private members in the base class are not available to its subclasses. Thus, within the Painting class, a reference like the following would not work. $abc = parent::name; // would not work within the Painting class If you want a member to be available to subclasses but not anywhere else, you can use the protected access modifier, which is shown in Figure 10.10. Inheriting Methods Every method defined in the base/parent class can be overridden when extending a class, by declaring a function with the same name. A simple example of overriding can be found in Listing 10.8 in which each subclass overrides the __toString() method. To access a public or protected method or property defined within a base class from within a subclass, you do so by refixing the member name with parent::. So to access the parent's __toString() method you would simply use parent::__toString().	[05]	CO 4	L 2
6(b)	Write a PHP program to create a class STUDENT with the following specification. Data members: Name, Roll number, Average marks Member function: Read(getters) and write (setters) Use the above specification to read and print the information of 2 students.	[05]	CO 4	L 3

```php
<?php
    class Student{
        private $name;
        public function setName($name,$rollnum,$avgmarks){
            $this->name = $name;
            $this->rollnum=$rollnum;
            $this->avgmarks=$avgmarks;
        }
        public function getName(){
            echo 'welcome '. $this->name."<br/>";
            echo 'Your Roll num is:'.$this->rollnum."<br/>";
            echo 'avgmarks:'.$this->avgmarks."<br/>";
            echo "_________________<br/>";
        }
    }
    $stu1 = new Student();
    $stu1->setName('xyz',12,89);
    $stu1->getName();
    $stu2 = new Student();
    $stu2->setName('abc',13,91);
    $stu2->getName();
?>
```

OUTPUT:

welcome xyz
Your Roll num is:12
avgmarks:89

_____
welcome abc
Your Roll num is:13
avgmarks:91

_____