IP Assignment - 2

Rushank Ghanshyam Sheta

TE - IT : 46

Q.1) Explain how session management is done in PHP. Clearly explain how to create, access, modify session variables in PHP.

Q.2) Explain the role of a cookie and differentiate it from sessions. Write a PHP script to check whether the cookie is set or not. Explain string functions in PHP.

Q.3) What are the characteristics of Rich Internet Application?

Q.4) Draw the diagram for AJAX Web application model and Traditional Web application model and compare them.

Q.5) Explain in detail JSON mash ups with a neat diagram.

Q.6) Draw the diagram for AJAX Web application model and Traditional Web application model and compare them.

Q1. PHP Session -Explain how session management is done in PHP. Clearly explain how to create, access, modify session variables in PHP.

- What is a Session?

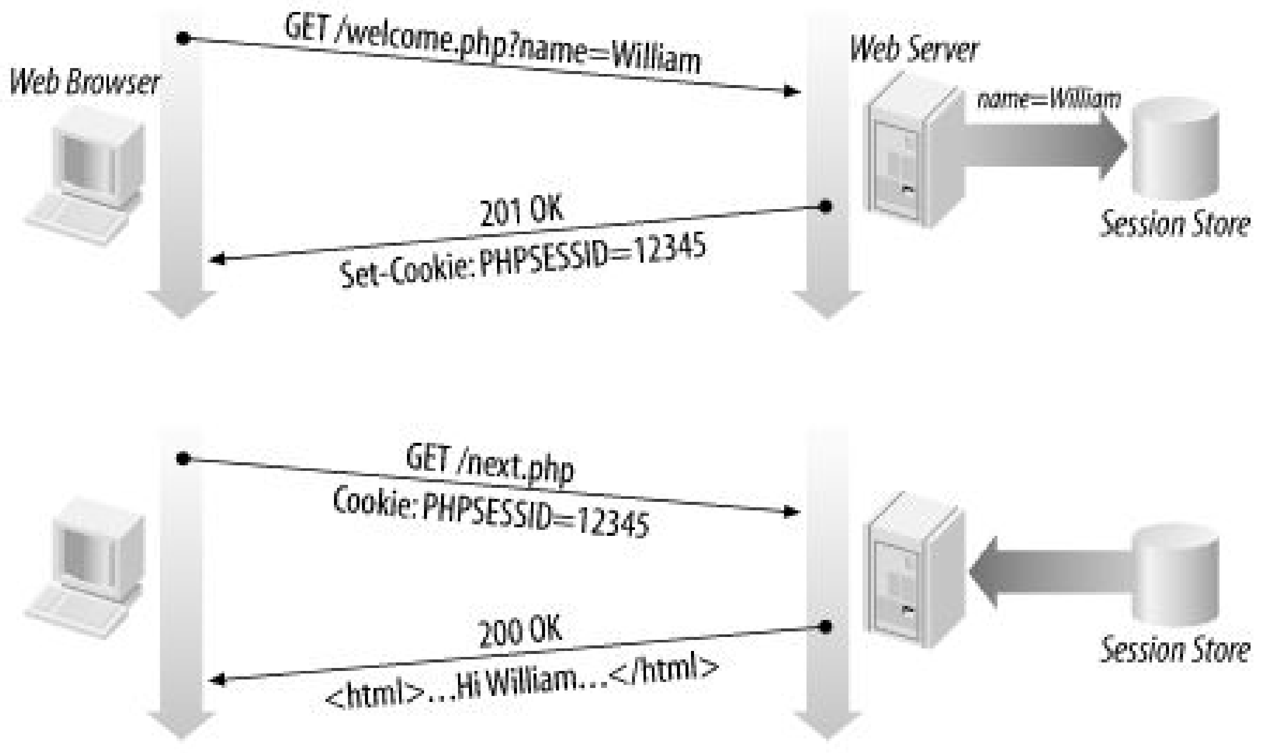
A **session** is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.
In **PHP**, a **session** provides a way to store web page visitor preferences on a web server in the form of variables that can be used across multiple pages. Unlike a cookie, variable information is not stored on the user's computer. The session sets a tiny cookie on the user's computer to serve as a key.

- How is Session Management done in PHP?

The out-of-the-box configuration of PHP session management uses disk-based files to store session variables. Using files as the session store is adequate for most applications in which the numbers of concurrent sessions are limited. A more scalable solution uses a MySQL database.

PHP provides a cookie-based implementation for session management. The `$_SESSION` array is used for storing session data. PHP automatically generates a session ID and sends a session cookie containing this session ID to the client machine. The PHP functions for session management are listed in the following table.



When a user first enters the session-based application by making a request to a page that starts a session, PHP generates a session ID and creates a file that stores the session related variables. PHP sets a cookie to hold the session ID in the response the script generates. The browser then records the cookie and includes it in subsequent requests. In the example shown in the above figure, the script welcome.php records session variables in the session store, and a request to next.php then has access to those variables because of the session ID.

- <u>Operations on Session</u>

1. `Start a PHP Session`

   The `session_start()` function first checks to see if a session already exists by looking for the presence of a session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

```php
<?php
// Starting session
session_start();
?>
```

2. `Modify`

   You can store all your session data as key-value pairs in the `$_SESSION[]` superglobal array. The stored data can be accessed during the lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

```php
<?php
// Starting session
session_start();

// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

3. `Access`

   To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling `session_start()` and then pass the corresponding key to the `$_SESSION` associative array.

```php
<?php
// Starting session
session_start();

// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

4. Delete

   If you want to remove certain session data, simply unset the corresponding key of the `$_SESSION` associative array, as shown in the following example:

```php
<?php
// Starting session
session_start();
// Removing session data
if(isset($_SESSION["lastname"])){
    unset($_SESSION["lastname"]);
}
?>
```

However, to destroy a session completely, simply call the `session_destroy()` function. This function does not need any argument and a single call destroys all the session data.

```php
<?php
// Starting session
session_start();
 // Destroying session
session_destroy();
?>
```

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of `session.gc_maxlifetime` variable in the PHP configuration file (`php.ini`).

- <u>PHP script to check if cookie is set or not</u>

```php
<!DOCTYPE html>
<?php
     setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
     if(count($_COOKIE) > 0) {
          echo "Cookies are enabled.";
     }
     else {
          echo "Cookies are disabled.";
     }
?>

</body>
</html>
```

Q2. Cookie -Explain the role of a cookie and differentiate it from sessions. Write a PHP script to check whether the cookie is set or not. Explain string functions in PHP.

- <u>What is a Cookie?</u>

A **cookie** is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the **cookie** too.
There are two **types of cookies** in PHP, they are: Session **Cookie**: This **type of cookies** are temporary and expire as soon as the session ends or the browser is closed. Persistent **Cookie**: To make a **cookie** persistent we must provide it with an expiration time.

- <u>Role of Cookie</u>

  A cookie is often used to identify a user.cookies allow us to track the state of the application using small files stored on the user's computer. The path were the cookies are stored depends on the browser. Personalizing the user experience – this is achieved by allowing users to select their preferences. The page requested that follow are personalized based on the set preferences in the cookies.

- <u>Role of Sessions</u>

  To store important information such as the user id more securely on the server where malicious users cannot temper with them. Sessions are used to pass values from one page to another.
  It is also used when you want the alternative to cookies on browsers that do not support cookies, to store global variables in an efficient and more secure way compared to passing them in the URL.

- <u>Cookie vs Session</u>

Cookie	**Session**
● Cookies are client-side files that contain user information	● Sessions are server-side files which contain user information
● Cookie ends depending on the lifetime you set for it	● A session ends when a user closes his browser
● You don't need to start cookie as it is stored in your local machine	● In PHP, before using $_SESSION, you have to write session_start(); Likewise for other languages

• The official maximum cookie size is 4KB	• Within-session you can store as much data as you like. The only limits you can reach is the maximum memory a script can consume at one time, which is 128MB by default
• A cookie is not dependent on Session	• A session is dependent on Cookie
• There is no function named unsetcookie()	• Session_destroy(); is used to destroy all registered data or to unset some

- <u>PHP String operations</u>

1. Length of String

```
echo strlen(“Hi This is a string operation”);
      # strlen(string) gives length of entered string.
>> 29
```

2. Number of words in a string

```
echo str_word_count(“Hi This is a string operation”);
      # no of words in a string is returned.
>> 6
```

3. Reversing a string

```
echo strrev(“Welcome”);
      # Reverses the entered string.
>> emocleW
```

4. Find Text in a string

```
echo strpos(“I am nobody”,”am”);
      # strpos(string, text), returns the position when text is found.
>> 2
```

5. Replacing text within a string

```
echo str_replace(“before”,”after”,”I am before.”);
      # str_replace(str_to_be_replaced,text,string), replaces string.
>> I am after.
```

6. Convert text into Title Case

```
echo ucwords(“This is a title”);
      #First alphabet of a word is always capital as in Title.
>> This Is A Title
```

7. Convert whole string to UPPERCASE

```
echo strtoupper(“Welcome”);
      # Whole string is converted to UPPERCASE.
>> WELCOME
```

8. Convert whole string to lowercase

```
echo strtolower(“WeLcOmE”);
      # Whole string is converted to lowercase.
>> welcome
```

9. Replete a given string for specified number of times

```
echo str_repeat(“=”,13);
      # str_repeat(string,repeat), repeats the string for repeat times.
>> =============
```

10. Print substring from a string

```
echo substr(“Hi This is ...”,1,8);
      # substr(string, start, end), returns substring.
>> i This i
```

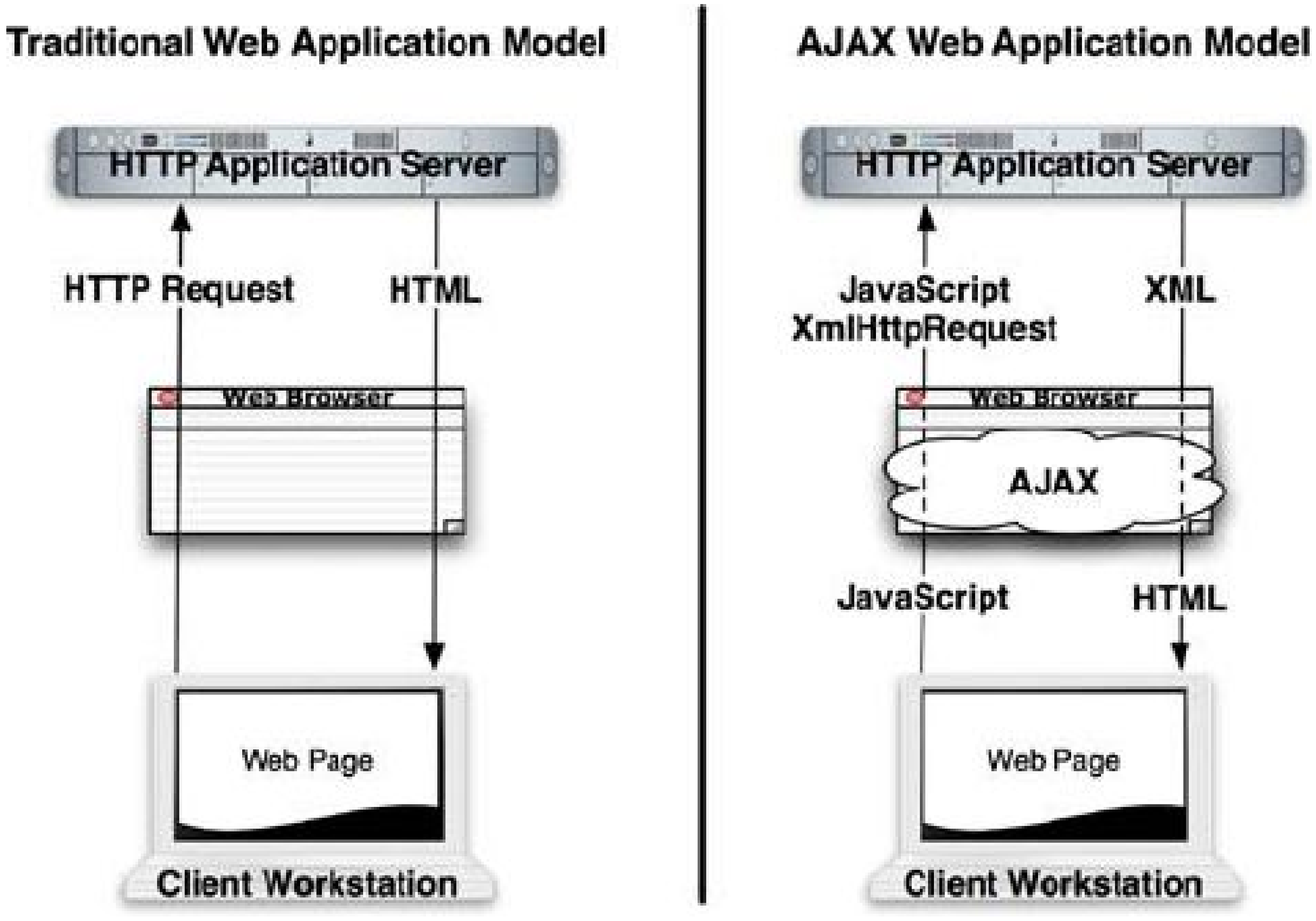
Q3. RIA -Characteristics of rich internet application

- **RIAs** are web applications that have most of the characteristics of desktop applications, typically delivered through web-browser **plug-ins** or independently via sandboxes or virtual machines.
- **RIAs** can run **faster** and be more engaging. They can offer users a better visual experience, better accessibility, usability and more interactivity than traditional browser applications that use only HTML and HTTP.
- A **RIA** can perform **computation** on both the client side and server side. User Interface, its related activity and capability on the client side and the data manipulation and operation on the application server side.
- **RIA** is **developed** using various technologies such as Java, Silverlight, JavaFX, JavaScript, REST/WS etc.

- Characteristics of RIA:
    - ***Performance*** - RIAs can often perform better than traditional applications on the basis of the characteristics of network and applications, performance of server also improved by offloading possible processing work to the client system and also perceived performance in terms of UI responsiveness and smoother visual transitions and animations are key aspects of any RIA.
    - ***Offline use*** - When connectivity is unavailable, it might still be possible to use an RIA. An RIA platform let the user work with the application without connecting to the Internet and synchronizing it automatically when the user goes live.
    - ***Consistency of look and feel*** - With RIA tools, the user interface and experience with different browsers, devices and operating systems can be more carefully controlled and made consistent.
    - ***Security*** - RIAs should be as secure as any other web application, and the framework should be well equipped to enforce limitations appropriately when the user lacks the required privileges, especially when running within a constrained environment such as a sandbox.
    - ***Advanced Communications*** - Sophisticated communications with supporting servers through optimized network protocols can considerably enhance the user experience.
    - ***Rapid Development*** - An RIA Framework should facilitate rapid development of a rich user experience through its easy-to-use interfaces in ways that help developers.
    - ***Direct Interaction*** - An RIA can use a wider range of controls that allow greater efficiency and enhance the user experience. In RIAs, for example, users can interact directly with page elements through editing or drag-and-drop tools. They can also do things like pan across a map or other image.
    - ***Better Feedback*** - Because of their ability to change parts of pages without reloading, RIAs can provide the user with fast and accurate feedback, real-time confirmation of actions and choices, and informative and detailed error messages.
    - ***Improved Features*** - RIA allows programmers to embed various functionalities in graphics-based web pages that look fascinating and engaging like desktop applications. RIA provides complex application screens on which various mixed media, including different fonts, vector graphic and bitmap files online conferencing etc. are paused by using different modern development tools.

- ***Partial-page updating*** - RIAs incorporate additional technologies, such as real-time streaming, high-performance client-side virtual machines, and local caching mechanisms that reduce latency (wait times) and increase responsiveness.

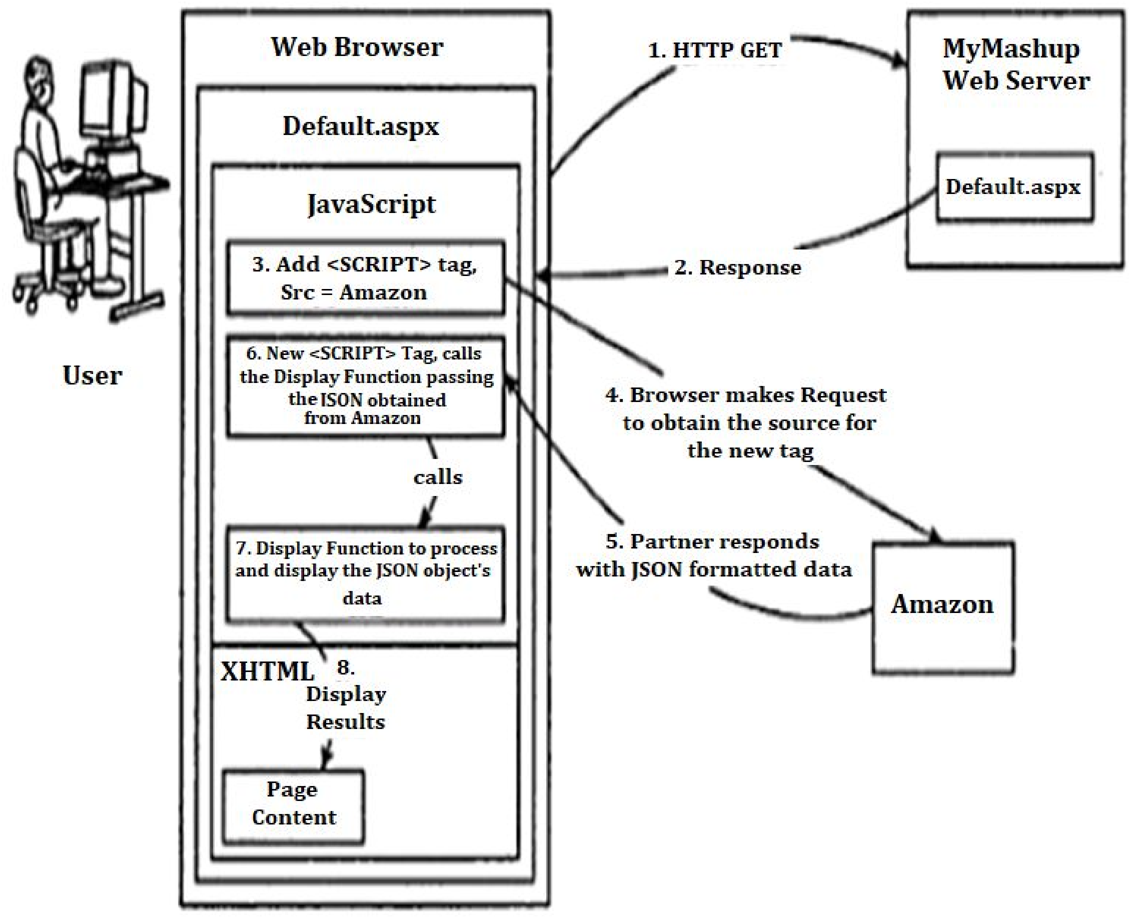
Q4. AJAX -Draw the diagram for AJAX Web application model and Traditional Web application model and compare them.



Traditional Web Application	AJAX Web Application
❏ Slower.	❏ Faster.
❏ Entire page is updated even for small	❏ Partial page update is possible using

changes in the entire page.	callback() function.
❏ Has Higher Traffic, since the entire page gets reloaded every time.	❏ Has Lower Traffic, because a part of parts of pages are updated.
❏ Has Higher Bandwidth consumption, since the entire page gets reloaded every time.	❏ Has Lower Bandwidth consumption, because a part of parts of pages are updated.
❏ Difficult to Navigate.	❏ Easier to Navigate.
❏ Is Less Compatible with other languages.	❏ Has Higher Compatibility with other laguages.
❏ Server communication is Synchronous.	❏ Server communication is Asynchronous.
❏ Slow and error prone form validation, because of higher bandwidth.	❏ Fast and proper form validation.
❏ Less Responsive.	❏ More Responsive.
❏ User Experience gets Interrupted because every time the page gets reloaded.	❏ User Experience is Uninterrupted.

Q5. JSON -JSON mashups with diagram


Web Browser
Default.aspx
JavaScript
3. Add <SCRIPT> tag, Src = Amazon
6. New <SCRIPT> Tag, calls the Display Function passing the JSON obtained from Amazon
calls
7. Display Function to process and display the JSON object's data
XHTML
8. Display Results
Page Content
User
1. HTTP GET
MyMashup Web Server
Default.aspx
2. Response
4. Browser makes Request to obtain the source for the new tag
5. Partner responds with JSON formatted data
Amazon

The readability and simplicity features of JSON have made it one of the most popular notations in the mashup community. JSON, being a notation, provides a way in which objects are written so that human beings can read it easily. It has built-in JavaScript features that made JSON a viable mashup technology. The architecture of mashups in JSON implementation is shown in the above figure.

The flow of JSON mashups that use dynamic Script method goes in the following steps:

1. The flow of the process starts with the browser sending requests to the server by using HTTP GET.
2. The Web Server responds with a page that includes the following couple of important JavaScript functions:
   a. A parsing function that expects JavaScript objects to be parameters.
   b. The Dynamic Script method is the core of the initiation script through which a new script tag is added to the page, specifying the source for that script tag to be the Uniform Resource Locator (URL) at some partner site.
3. The source code for the new script tag gets loaded by the browser.

4. Amazon receives an HTTP Get request sent from the browser using the loaded script.
5. A JavaScript object, after being serialized into a JSON Object, is served by the partner site.
6. A function call to the render function wraps the JSON script, and the JavaScript entirely becomes the content for the script tag.
7. The new piece of JavaScript is tried for execution by the browser, which calls the render method from step 2(a).
8. The server invokes the render method and evaluates the JSON script, which is converted into a JavaScript object. The data contained in the render method is pushed into the page after the render method, which uses the new JavaScript object in its execution

---

END