AAPP009-4-2 Web Development

PHP Cookie & PHP Session

Topic & Structure

- Overview of Cookies and Sessions
- Cookies
  - Setting Cookies in PHP
  - Reading Cookies
  - Accessing with $_COOKIE
  - Deleting Cookies
- Sessions
  - Starting a Session
  - Storing Session Data
  - Using $_SESSION superglobal
  - Destroying a Session
- Implementing Login System with Sessions
- Remember Me Functionality with Cookies

Learning Outcomes

- At the end of this lecture, you should be able to:
  - Explain what cookies and sessions are and their importance in web development.
  - Implement Cookies in PHP
  - Manage Sessions in PHP

Introduction

- Sessions and cookies are tools used in PHP to store data that needs to be available across different pages of a website.
- They allow you to keep track of user information and preferences, making your web app more interactive and personalized.

Cookies

- A cookie is a small file that the server places on the user's computer.
- Commonly used to identify a user on a website.
- Each time the same computer requests a page with a browser, it sends the cookie along with the request.
- PHP has the capability to both create and retrieve cookie values, making it a versatile tool for managing user data and preferences.

Structure of a Cookie

- A cookie is created in PHP using the setcookie() function.
- Function Syntax: The syntax for the setcookie() function is as follows: setcookie(name, value, expire, path, domain, secure, httponly);

Parameter	Description	Required
name	This is the only required parameter. It sets the name of the cookie.	Yes
value	This optional parameter sets the value of the cookie.	No
expire	This optional parameter sets the expiration date of the cookie.	No
path	This optional parameter specifies the server path of the cookie.	No
domain	This optional parameter sets the domain of the cookie.	No
secure	This optional parameter indicates whether the cookie should only be transmitted over a secure HTTPS c onnection.	No
httponly	This optional parameter indicates whether the cookie is accessible only through the HTTP protocol and not through scripting languages like JavaScript.	No

Usage of time() for Expire Parameter

- time() function returns the current timestamp, which is the number of seconds since January 1, 1970.
- When setting a cookie, we often need to specify an expiration time, which is done using a future timestamp.
- time() + (86400 * 30)
  - time(): Returns the current timestamp.
  - 86400: Represents the number of seconds in one day (24 hours * 60 minutes * 60 seconds).
  - 86400 * 30: Calculates the number of seconds in 30 days.
- So, time() + (86400 * 30) = adds 30 days' worth of seconds to the current timestamp, resulting in a future timestamp 30 days from now.

Writing Cookies in PHP

```php
// Set a cookie
setcookie('user', 'JohnDoe', time() + (86400 * 30), "/", "example.com", true, true);

// Access cookie data
echo $_COOKIE['user'];
```

- name is set to 'user'.
- value is set to 'JohnDoe'.
- expire is set to time() + (86400 * 30), which means the cookie will expire in 30 days.
- path is set to '/', making the cookie available across the entire domain.
- domain is set to 'example.com'.
- secure is set to true, meaning the cookie will only be sent over HTTPS.
- httponly is set to true, making the cookie inaccessible to JavaScript.

Reading a Cookie

- To read a cookie in PHP, use the $_COOKIE superglobal array.

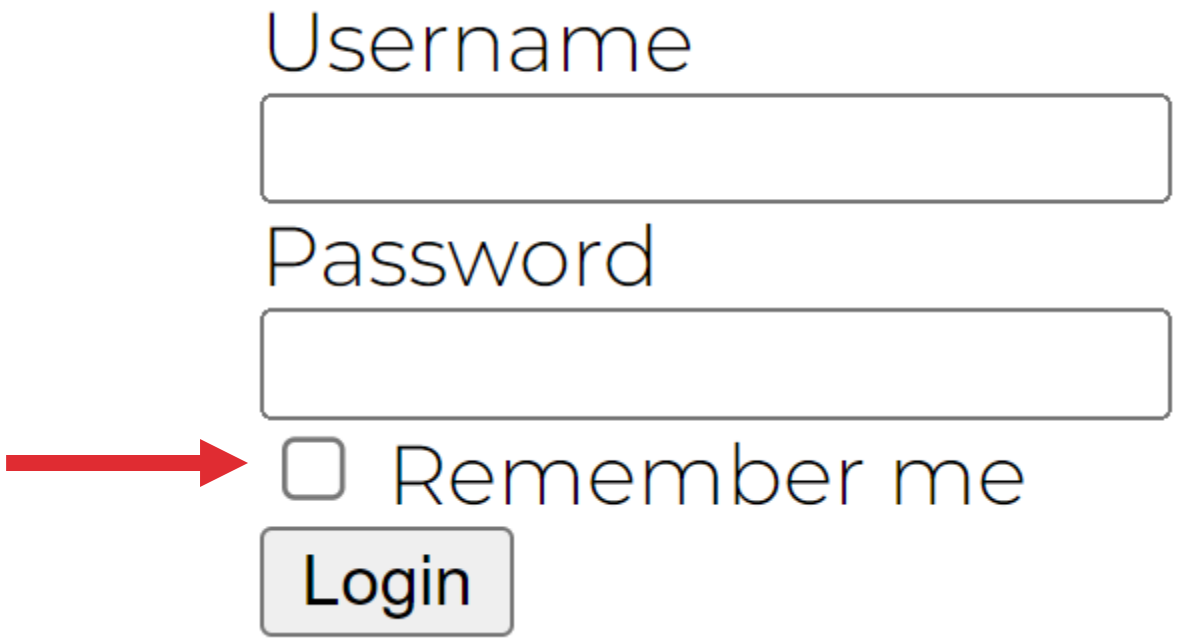
```php
// Check if the cookie is set
if(isset($_COOKIE['user'])) {
    // Retrieve the value of the cookie
    $user = $_COOKIE['user'];
    echo "User: " . $user;
} else {
    echo "Cookie 'user' is not set!";
}
```

  - The isset() function checks if the cookie named 'user' exists.
  - If it does, the value of the cookie is retrieved and stored in the $user variable.
  - The value is then printed out.

Common Uses of Cookies

- User Preferences
  - Remembering user settings like language, theme, or layout preferences.
  - Example: Storing a user's preferred language to display the website in that language on subsequent visits.
- Session Management
  - Keeping users logged in by storing session identifiers.
  - Example: Storing a session ID to maintain a user's login state across different pages.
- Tracking and Analytics
  - Tracking user behavior for analytics purposes.
  - Example: Storing a unique identifier to track user visits and interactions on the website.
- Personalization
  - Providing personalized content based on user behavior and preferences.
  - Example: Showing personalized recommendations based on past browsing history.

Common Usage



Will demonstrate this usage after PHP Session

Sessions

- Store information across multiple pages.
- Useful for applications that need to remember user data.
- Without sessions, data would be lost with each new page load.

How to Use Sessions in PHP

1. Start a Session:
   - Must be at the beginning of your code, before any HTML.

```
session_start();
```

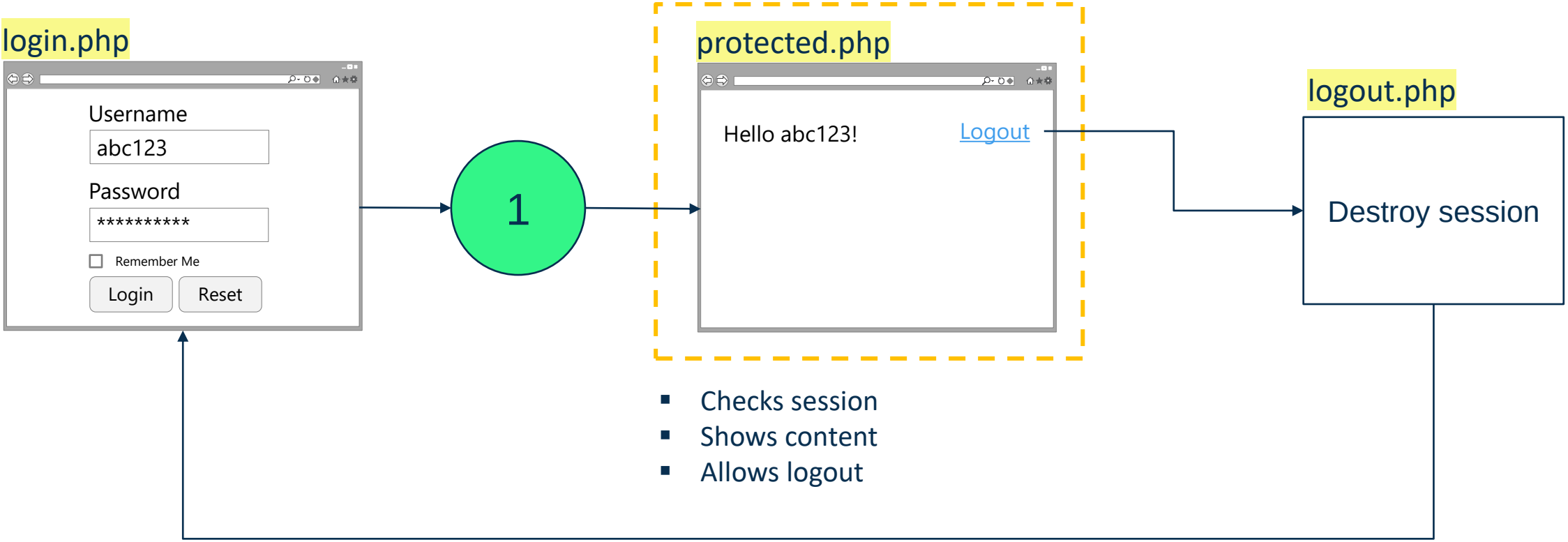
2. Store Data:
   - Use the $_SESSION associative array.

```
$_SESSION['username'] = 'JohnDoe';
```

3. Retrieve Data:
   - Access stored session data.

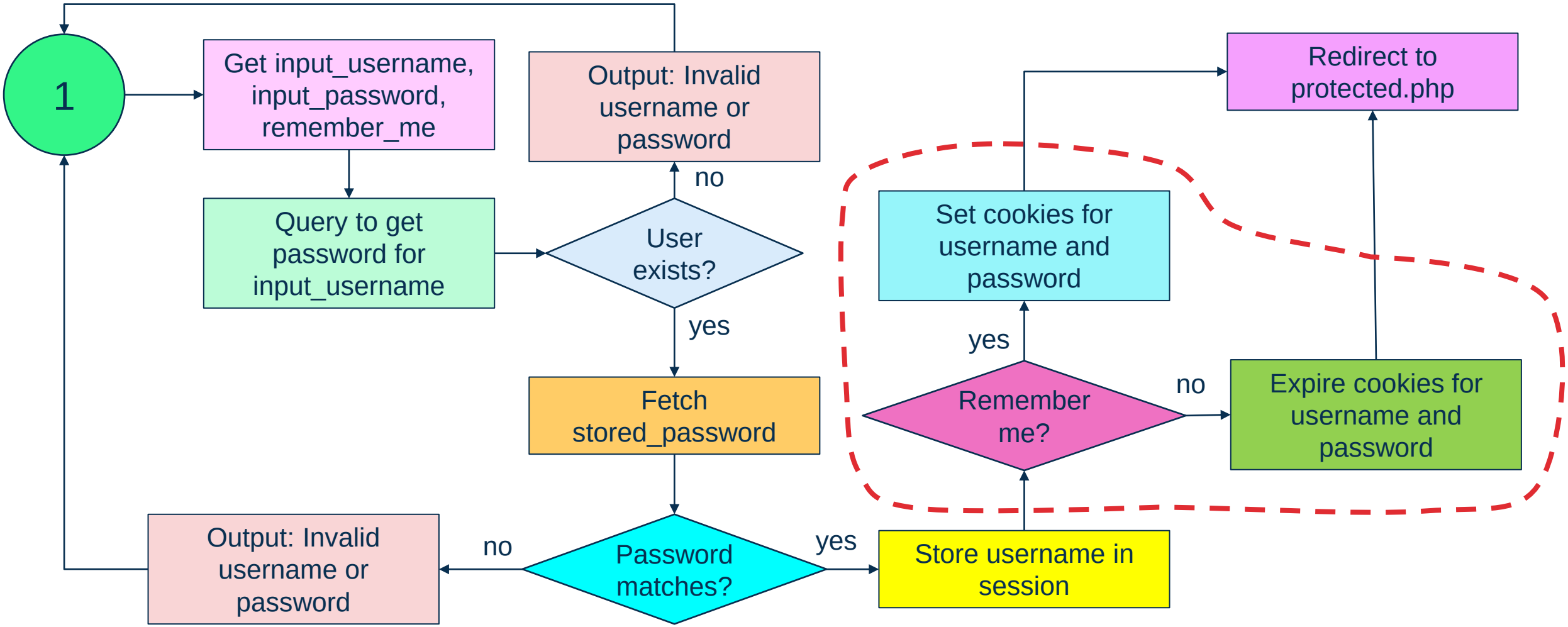
```
echo $_SESSION['username'];
```

Login Process



- Checks session
- Shows content
- Allows logout

Login and Remember Me Logic



login.php

```
<form method="post">
    <h1>User Login</h1><br>
    Username<br>
    <input type="text" name="username" value="<?php echo isset($_COOKIE['username']) ?
    $_COOKIE['username'] : ''; ?>" required><br><br>

    Password<br>
    <input type="password" name="password" value="<?php echo isset($_COOKIE['password']) ?
    $_COOKIE['password'] : ''; ?>" required><br><br>

    <input type="checkbox" name="remember_me" <?php echo isset($_COOKIE['username']) ? 'checked' : '';
    ?>> Remember me<br><br>

    <input type="submit" value="Login">
    <input type="reset" value="Reset">
</form>
```

To check if a cookie named username exists and print its value if it does; otherwise, print an empty string (using ternary operator).

(condition) ? (true_expression) : (false_expression);

Login and Remember Me

```php
<?php
// Start the session
session_start();

// Check if form is submitted
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $input_username = $_POST['username'];
    $input_password = $_POST['password'];
    $remember_me = isset($_POST['remember_me']);

    // Query to get the password for the given username
    $sql = "SELECT password FROM users WHERE username = '$input_username'";
    $result = mysqli_query($conn, $sql);

    // Check if user exists
    if (mysqli_num_rows($result) > 0) {
        $row = mysqli_fetch_assoc($result);
        $stored_password = $row['password'];

        // Verify password
        if ($input_password == $stored_password) {
            // Store user information in session
            $_SESSION['username'] = $input_username;
```

```php
            // Set or clear cookies based on the checkbox
            if ($remember_me) {
                setcookie('username', $input_username, time() + (86400 * 30), "/"); // 30 days
                setcookie('password', $input_password, time() + (86400 * 30), "/"); // 30 days
            } else {
                setcookie('username', '', time() - 3600, "/"); // Expire the cookie
                setcookie('password', '', time() - 3600, "/"); // Expire the cookie
            }

            header("location:protected.php");
        } else {
            echo "<script>alert('Invalid username or password!');</script>";
        }
    } else {
        echo "<script>alert('Invalid username or password!');</script>";
    }

    mysqli_free_result($result);
}

mysqli_close($conn);
?>
```

protected.php

```php
<?php
// Start the session
session_start();

// Check if a session variable is set
if (!isset($_SESSION['username'])) {
    echo "<script>alert('Please login!');window.location.href='login.php';</script>";
}
?>
```

To check if a session variable named username is set.

- **$_SESSION**: Superglobal array in PHP used to store session variables.
- **isset()**: PHP built-in function that checks if a variable is set and is not null.
- **! (Not Operator)**: checks if the variable is not set.

```php
<?php
        // Shows content (i.e., the current user who is logged in)
        echo "<h1>Welcome to the protected page, " . $_SESSION['username'] . "!</h1>";
?>
```

logout.php

- session_start():
  - Initializes a session or resumes the current one based on a session identifier
  - Must be called before any session manipulation functions

- session_destroy():
  - Destroys all data registered to a session.
  - Requires an active session to be initialized with session_start().

- Cannot use session_destroy() without first calling session_start()

```php
<?php
// Start the session
session_start();

// Destroy the session
session_destroy();

echo "<script>alert('You have been logged out.');window.location.href='login.php';</script>";
?>
```

Summary

Sessions	Cookies
• Stored on the server. • Use the $_SESSION superglobal array to store and access data. • Data is available across multiple pages during a user's visit. • Example: Storing a user's login status.	• Stored on the client's browser. • Use the $_COOKIE superglobal array to retrieve data. • Data is sent with every HTTP request from the client. • Example: Remembering a user's language preference.

Q & A