Using Cookies in PHP

Overview

Cookies are small pieces of data stored on the client's browser. They are commonly used to track user activity, store preferences, or maintain session information.

Key Concepts

- **Creating Cookies**: Use the `setcookie()` function.
- **Accessing Cookies**: Use the `$_COOKIE` superglobal.
- **Cookie Expiry**: Set an expiration time (in seconds) using the `time()` function.
- **Deleting Cookies**: Set the expiration time to a past date.

Practical Examples

Example 1: Setting a Cookie

```
<?php
// Set a cookie with a name "user" and value "John Doe"
setcookie("user", "John Doe", time() + 3600, "/"); // Expires in 1 hour
echo "Cookie 'user' is set.";
?>
```

Example 2: Accessing a Cookie

```
<?php
if (isset($_COOKIE['user'])) {
    echo "Welcome " . $_COOKIE['user'];
} else {
    echo "Cookie 'user' is not set.";
}
?>
```

Example 3: Deleting a Cookie

```
<?php
// Delete the "user" cookie by setting its expiration to the past
setcookie("user", "", time() - 3600, "/");
echo "Cookie 'user' is deleted.";
?>
```

Practical Questions

1. Create a PHP script to set a cookie with the user's favorite color and display it.
2. Write a script to check if a cookie named "visits" exists. If it does, increment its value; otherwise, set it to 1.
3. Delete the "visits" cookie after displaying its value.

HTTP Authentication

Overview

HTTP Authentication is a method to restrict access to web pages. It uses a username and password to authenticate users.

Key Concepts

- **Basic Authentication**: Uses the `$_SERVER['PHP_AUTH_USER']` and `$_SERVER['PHP_AUTH_PW']` superglobals.
- **Realm**: A string that defines the protected area.
- **Unauthorized Access**: Send a `401 Unauthorized` header to prompt the user for credentials.

Practical Examples

Example 1: Basic HTTP Authentication

```
<?php
$username = "admin";
$password = "password";

if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Unauthorized';
    exit;
} else {
    if ($_SERVER['PHP_AUTH_USER'] == $username && $_SERVER['PHP_AUTH_PW'] == $password) {
        echo "Welcome Admin!";
    } else {
        echo "Invalid username or password.";
    }
}
?>
```

Practical Questions

1. Implement HTTP authentication for a page that only allows access if the username is "user" and the password is "secret".
2. Modify the script to display a custom message if the user cancels the authentication prompt.

Using Sessions

Overview

Sessions are used to store user-specific data on the server. Each session is identified by a unique session ID, which is stored in a cookie on the client's browser.

Key Concepts

- **Starting a Session**: Use `session_start()`.
- **Storing Session Data**: Use the `$_SESSION` superglobal.
- **Destroying a Session**: Use `session_destroy()`.

Practical Examples

Example 1: Starting a Session and Storing Data

```
<?php
session_start();
$_SESSION['username'] = "John Doe";
echo "Session username: " . $_SESSION['username'];
?>
```

Example 2: Accessing Session Data

```php
<?php
session_start();
if (isset($_SESSION['username'])) {
    echo "Welcome back, " . $_SESSION['username'];
} else {
    echo "Session not set.";
}
?>
```

Example 3: Destroying a Session

```php
<?php
session_start();
session_destroy();
echo "Session destroyed.";
?>
```

Practical Questions

1. Create a PHP script to store a user's login status in a session and display it.
2. Write a script to destroy a session after displaying its data.
3. Implement a login system using sessions to restrict access to a page.

Summary of Key Points

- **Cookies**: Store data on the client's browser. Use `setcookie()` and `$_COOKIE`.
- **HTTP Authentication**: Restrict access using a username and password. Use `$_SERVER['PHP_AUTH_USER']` and `$_SERVER['PHP_AUTH_PW']`.
- **Sessions**: Store data on the server. Use `session_start()`, `$_SESSION`, and `session_destroy()`.