PHP Solution 9-1: A Simple Session Example

This should take only a few minutes to build, but you can also find the complete code in `session_01.php`, `session_02.php`, and `session_03.php`, in the `ch09` folder.

1. Create a page called `session_01.php` in a new folder called `sessions` in the `phpsols` site root. Insert a form with a text field called `name` and a `Submit` button. Set the `method` to `post` and `action` to `session_02.php`. The form should look like this:

```
<form method="post" action="session_02.php">
    <p>
        <label for="name">Name:</label>
        <input type="text" name="name" id="name">
    </p>
    <p>
        <input type="submit" name="Submit" value="Submit">
    </p>
</form>
```

2. In another page called `session_02.php`, insert this above the `DOCTYPE` declaration:

```
<?php
// initiate session
session_start();
// check that form has been submitted and that name is not empty
if ($_POST && !empty($_POST['name'])) {
    // set session variable
    $_SESSION['name'] = $_POST['name'];
}
?>
```

   The inline comments explain what's going on. The session is started, and as long as `$_POST['name']` isn't empty, its value is assigned to `$_SESSION['name']`.

3. Insert the following code between the `<body>` tags in `session_02.php`:

```
<?php
// check session variable is set
if (isset($_SESSION['name'])) {
    // if set, greet by name
    echo 'Hi, ' . $_SESSION['name'] . '. <a href="session_03.php">Next</a>';
} else {
    // if not set, send back to login
    echo 'Who are you? <a href="session_01.php">Login</a>';
}
?>
```

   If `$_SESSION['name']` has been set, a welcome message is displayed along with a link to `session_03.php`. Otherwise, the page tells the visitor that it doesn't recognize who's trying to gain access and provides a link back to the first page.

Caution Take care when typing the following line:

```
echo 'Hi, ' . $_SESSION['name'] . '. <a
href="session03.php">Next</a>';
```

The first two periods (surrounding `$_SESSION['name']`) are the PHP concatenation operator. The third period (immediately after a single quote) is an ordinary period that will

be displayed as part of the string.

4. Create `session_03.php`. Type the following above the `DOCTYPE` to initiate the session:

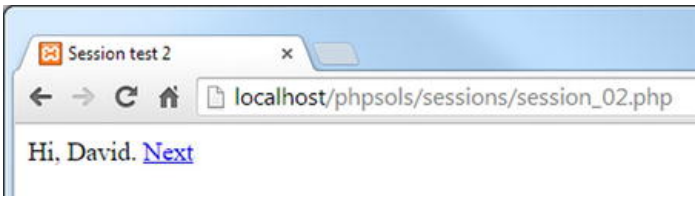
```
<?php session_start(); ?>
```

5. Insert the following code between the `<body>` tags of `session_03.php`:

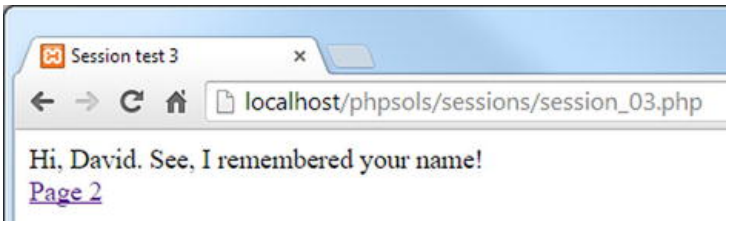
```
<?php
// check whether session variable is set
if (isset($_SESSION['name'])) {
    // if set, greet by name
    echo 'Hi, ' . $_SESSION['name'] . '. See, I remembered your name!<br>';
    // unset session variable
    unset($_SESSION['name']);
    // invalidate the session cookie
    if (isset($_COOKIE[session_name()])) {
        setcookie(session_name(), '', time()-86400, '/');
    }
    // end session
    session_destroy();
    echo '<a href="session_02.php">Page 2</a>';
} else {
    // display if not recognized
    echo "Sorry, I don't know you.<br>";
    echo '<a href="session_01.php">Login</a>';
}
?>
```

   If `$_SESSION['name']` has been set, the page displays it, then unsets it and invalidates the current session cookie. By placing `session_destroy()` at the end of the first code block, the session and its associated variables cease to be available.

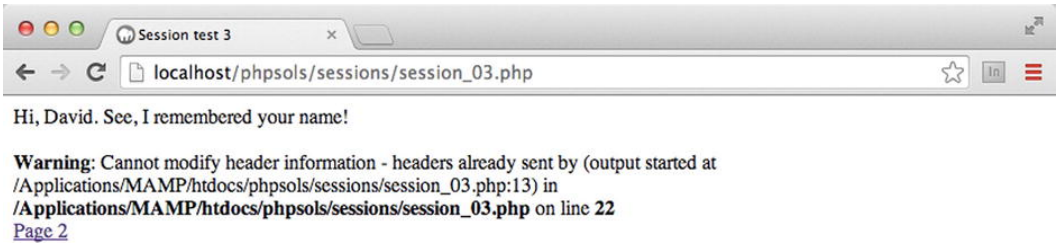
6. Load `session_01.php` into a browser, type your name in the text field, and click **Submit**.
7. You should see something like the following screenshot. At this stage, there is no apparent difference between what happens here and in an ordinary form.



8. When you click **Next**, the power of sessions begins to show. The page remembers your name, even though the `$_POST` array is no longer available to it. If you're using XAMPP as your testing setup, you'll probably see something similar to the following screenshot.

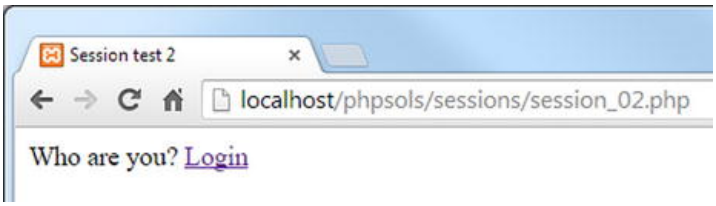


   However, with other setups, such as MAMP, you're likely to get a "**headers already sent**" warning message like this:

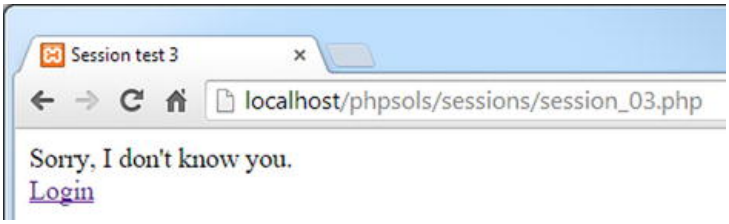


Note As explained in Chapter 4, XAMPP doesn't produce the warning about headers because it's configured to buffer the first 4 KB of output. However, not all servers buffer output, so it's important to fix this problem.

9. Click the link to Page 2 (if you got an error message, it's just below the message). The session has been destroyed, so this time `session_02.php` has no idea who you are.



10. Type the address of `session_03`.php in the browser address bar and load it. It, too, has no recollection of the session and displays an appropriate message.



Even if you didn't get the warning message in step 8, you need to prevent it from happening when you deploy pages that rely on sessions to other servers. The error message not only looks bad, but it also means `setcookie()` can't invalidate the session cookie. Even though `session_start()` comes immediately after the opening PHP tag in `session_03.php`, the warning message is triggered by the `DOCTYPE` declaration, the `<head>`, and other HTML being output before `setcookie()`.