How to set and get session variables

Once you start a session, you can set and get session variables. To do that, you can use the global $_SESSION variable as shown in figure 12-6. This variable is an associative array that stores the data for the session.

In this figure, the examples use the $_SESSION array to get variables directly rather than using a function like the filter_input() function. That's because session data is kept on the server and thus can't be manipulated by the user.

The first example shows how to set a scalar value in the $_SESSION array. Then, the second example show how to get that value from the $_SESSION array.

After showing how to work with scalar values in a session, this figure shows how to work with an array in a session. Here, the first example uses the isset() function to check whether an element named cart has already been set in the $_SESSION array. If not, it sets that element to an empty array. Then, the second example sets two values in the cart array, and the third example gets the cart array and sends a list of its contents to the browser.

After showing how to store an array in a session, this figure shows how to remove elements from a session. Here, the first example shows how to use the unset() function to remove an element from the $_SESSION array. Then, the second example shows how to remove all elements from the $_SESSION array by setting it to an empty array. When removing all elements from the $_SESSION array, don't use the unset() function directly on the $_SESSION variable as this can cause unpredictable results in your application.

As your application runs, PHP stores the $_SESSION array in memory. When the session ends, PHP saves the contents of the $_SESSION array in a file on the web server. Finally, PHP deletes this session data when the session expires. By default, a session expires after 24 minutes of inactivity.

When working with sessions, it's safe to store strings, numbers, and Boolean values in the session. In addition, it's safe to store arrays of these types of values. However, it often isn't safe to store objects in a session. For more information about objects, please refer to chapter 14.

The problem is that PHP stores the session data in a file on the web server. To do this for an object, PHP has to be able to convert the object to a text representation for storage. Then, when PHP reloads the session, it has to convert the text representation back to an object. These processes are known respectively as *serialization* and *deserialization*.

As a result, before you store an object in a session, you should consider three issues. First, your code needs to load the class definition before it calls the session_start() function. Second, objects that refer back to themselves can't be stored as text representations. Third, serialization is slow and takes up more storage space than the original object. For large scale web applications, there is a significant performance penalty for storing objects in a session.

Fortunately, an object often represents a row in a database. As a result, you can often store the primary key for the row in the session. Then, you can reload the data from the database into the object when the new page is loaded. This is

How to set and get scalar variables

Set a variable in a session

```
$_SESSION['product_code'] = 'MBT-1753';
```

Get a variable from a session

```
$product_code = $_SESSION['product_code'];
```

How to set and get arrays

Set an array in a session

```
if (!isset($_SESSION['cart'])) {
    $_SESSION['cart'] = array();
}
```

Add an element to an array that's stored in a session

```
$_SESSION['cart']['key1'] = 'value1';
$_SESSION['cart']['key2'] = 'value2';
```

Get and use an array that's stored in a session

```
$cart = $_SESSION['cart'];
foreach ($cart as $item) {
    echo '<li>' . $item . '</li>';
}
```

How to remove variables from a session

Remove a session variable

```
unset($_SESSION['cart']);
```

Remove all session variables

```
$_SESSION = array();
```

Description

- Once you start a session, you can use the superglobal $_SESSION variable to set and get the user's data for a session. This variable is an associative array.
- If necessary, you can use the isset() function to test if an element already exists in the $_SESSION array.
- You can use the unset() function to remove an element from the $_SESSION array. However, don't use the unset() function on the $_SESSION array itself as it can cause unpredictable results.
- You can set the $_SESSION array to an empty array to remove its contents.