One login process must allow the users to verify all portions of the application. Each portion of the application needs access to common properties (such as user ID and password) that have been set by the authentication tier for verification of valid access and valid levels of access. PHP provides the ability to store information for an application in server memory by declaring a *session*. A session is considered to include the complete interaction of the user with the application (such as the complete processes of transferring money from a saving account to a checking account). A session can be established as soon as the user signs in to the application. The session can be closed after the user logs out of the system (or the application times out, or is closed). When a session is closed, all properties stored in the memory of the server are removed by the garbage collector.

While the session is active, properties can be stored and shared throughout the application. Using this process, the user ID and password can be stored in session properties. Each part of the application can then verify that the user has logged in by determining if there are values in the `userid` and `password` properties.

Before you look at the login authentication process, let's look at how to determine if a user has logged into the system. The examples in this chapter do not include verifications of security access levels. However, the process to determine these levels would use similar code as shown in these examples.

Programming note—The `session_start` method call must be the first statement at the top of the code. There must be no spaces or code between the `<?php` tag and `session_start`. The `session_start` method produces an HTML header that would not be formed correctly if any code exists before the method call.

```
<?php
session_start();
if ((!isset($_SESSION['username'])) || (!isset($_SESSION['password']))) {
echo "You must login to access the ABC Canine Shelter Reservation System";
echo "<p>";
echo "<a href='login.php'>Login</a> | <a href='register.php'>Create an account</a>";
echo "</p>";
}
else {
echo "<p>Welcome back, " . $_SESSION['username'] . "</p>";
}
?>
```

Each program in the interface tier (that the user can access) would include code similar to the previous example. The `session_start` method, in this example, lets the operating system know that this program is part of an existing session (which is declared in the authentication tier). Each session is identified by the system using a uniquely generated ID. As long as the user (or system) has not closed the session, the session ID will be attached to any program, called by the user, which includes the `session_start` method. This allows the program to access all properties related to the current session.

The PHP `isset` method (in an `if` statement) can determine if values exist in the `username` and `password` properties. If values do not exist, it indicates that the user has not been authenticated. Session properties are retrieved (and set) using `$_SESSION`. In the previous example, if either of the properties is not set, the user is provided links to the login page (`login.php`) or the register page (`register.php`). If both properties are set, the user is welcomed to the system. The user has no choice but to log in to access the program. In addition, as mentioned in previous chapters, for more secure programs, the IP address of the user's machine, and the calling program can be determined to provide extra assurance that the user is authorized.