This example code will pull the information from a table in the database (`Users`) and place the information in an associate array (via the `mysqli_fetch_assoc` method). If the fields in the table are the same as the tag names from the XML file (`userid` and `password`), the associate array built will be similar to the array built using the code from Example 7-2. All previous code "`// ...... code to verify userid and password ....`" is replaced by the code shown here. No code below the statement should need adjusting.

Do It

1. Create a conversion program to determine the encrypted version of a password using the PHP method `password_hash` (see `http://php.net/manual/en/function.password-hash.php`).
2. Download the example files from this section. Add XML records to the `uidpass` file that could be used for access permissions (read only or read/write) and levels (user or administrator). Test to verify that the new `uidpass` file works correctly with the existing code. Make any necessary code changes to make it compatible.
3. Download the example files from this section. Add code to the program to limit attempts to log in with a bad password to three. Record any invalid attempt to log in (after three tries) in the user log.

Registration

In addition to authorizing user logons, most systems allow the users to create their own user IDs and passwords. By default, the self-created IDs are given the lowest priorities. An administrator can then go in and increase the level of privileges once the ID has been created. Some sites allow non-registered users (users who are not logged in).

Non-registered users should only be given read-only access to non-privileged information. Non-registered users are much more of a security risk because it would be difficult to determine their identity during a security breach. PHP provides the ability to retrieve the user's IP address using `$_SERVER['REMOTE_ADDR']`. This might provide some ability to trace the user. However, many users use free public access points, which generate random IP addresses. If the one of these points is used, it will be much more difficult to track them down.

In addition, providing the opportunity for users to create user IDs and passwords allows the program to gather additional information (such as name and e-mail) that can help with security, and also provide easy ability to promote the web site to users of the site. The success rate of selling what is offered on the site will be much higher to customers who are already familiar with the site. To encourage users to create user IDs and passwords, the site should offer them some benefit (such as access to the help desk) that is not available to visitors.

The *registration page* will, in many ways, work similarly to the login page. However, any valid user ID or password entered will be stored (instead of compared). Because the application will be updating the list of valid IDs, the application must validate the information before it is updated. You can use some of the techniques that you used previously when validating the `dog` class properties. First, as the user enters a user ID and password (which must pass the HTML5 validation shown previously), it is passed to a PHP program. Even though it is considered to be secure inside a session, the information will travel from the HTML form to the PHP code. The information should be validated again.