Form Processing and Business Logic

- XHTML forms enable web pages to collect data from users and send it to a web server for processing.

- Interaction of this kind between users and web servers is vital to e-commerce applications. Such capabilities allow users to purchase products, request information, send and receive web-based email, perform on-line paging and take advantage of various other online services.

- The XHTML document on the next few pages collects information from a user for the purposes of adding them to a mailing list.

- The PHP file on page 3 validates the data entered by the user through the form and “registers” them in the mailing list database.
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Sample form to take user input in XHTML</title>
  </head>
  <body>
    <h1>This is a sample registration form.</h1>
    Please fill in all fields and click Register.
    <form method="post" action="form.php">
      <img src="images/user.gif" alt="User" /> <br />
      Please fill out the fields below. <br />
      <img src="images/fname.gif" alt="First Name" />
      <input type="text" name="fname" />
    </form>
  </body>
</html>
<input type = "text" name = "lname" /><br />
Which operating system are you currently using?

- [ ] Windows XP
- [ ] Windows 2000
- [ ] Windows 98
- [ ] Linux
- [ ] Other

Create a submit button

<input type="submit" value="Register" />

</form>

</body>
</html>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- form.php -->
<!-- Read information sent from form.html -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Form Validation</title>
</head>
<body style = "font-family: arial,sans-serif">

<?php
extract($_POST);
// determine whether phone number is valid and print an error message if not
if ( !ereg( "^\([0-9]{3}\)\([0-9]{3}\)[0-9]{4}$", $phone ) ) {
    print( "<p><span style = "color: red; font-size: 2em">INVALID PHONE NUMBER:</span><br />
    A valid phone number must be in the form<br />
    <strong>(555)555-5555</strong><br />
    Click the Back button, enter a valid phone number and resubmit.<br />
    Thank You.</p>");
    die(); // terminate script execution
    
} ?>

Function extract (associativeArray) creates a variable-value pair corresponding to each key-value pair in the associative array $_POST.

See page 17 for explanation of regular expressions.

Function die() terminates script execution.
An error has occurred, no need to continue.
<p>Hi</p>
<p><span style="color: blue"> <strong><?php print( "$fname" ); ?></strong> </span>. Thank you for completing the survey.<br />
You have been added to the <span style="color: blue"> <strong><?php print( "$magazine " ); ?></strong> </span> mailing list.</p>
<p>The following information has been saved in our database:<br />
<table border="0" cellpadding="0" cellspacing="10">
<tr>
<td bgcolor="#ffffaa">Name</td>
<td bgcolor="#ffffbb">Email</td>
<td bgcolor="#ffffcc">Phone</td>
<td bgcolor="#ffffdd">OS</td>
</tr>
<?php
// print each form field's value
print( "<td>$fname $lname</td> <td>$email</td> <td>$phone</td> <td>$os</td>" );
?>
</tr>
</table>
</p>
<p>This is only a sample form. You have not been added to a mailing list.</p>
This is a sample registration form.

Please fill in all fields and click Register.

**User Information**

Please fill out the fields below.

- **First Name**
- **Last Name**
- **Email**
- **Phone**

Must be in the form (555)555-5555

**Publications**

Which publication would you like information about?

- **Velo-News**

**Operating System**

Which operating system are you currently using?

- Windows Vista
- Windows XP
- Windows 2000
- Linux
- Other

Register
After execution of form.php has verified correct entries made within the form.
Hi Mark. Thank you for completing the survey.
You have been added to the Pro Cycling mailing list.

The following information has been saved in our database:
Name            Email            Phone       OS
Mark Llewellyn  markl@cs.ucf.edu  (407)823-2790  Windows Vista

This is only a sample form. You have not been added to a mailing list.
User enters an improperly formatted telephone number in the form.
INVALID PHONE NUMBER:
A valid phone number must be in the form (555)555-5555
Click the Back button, enter a valid phone number and resubmit.
Thank You.
How the Form Example Works

• The action attribute of the form element, indicates that when the user clicks the Register button, the form data will be posted to form.php for processing.

• Using method = “post” appends the form data to the browser request that contains the protocol (i.e., HTTP) and the requested resource’s URL. Scripts located on the web server’s machine (or accessible through the network) can access the form data sent as part of the request.

• Each of the form’s input fields are assigned a unique name. When Register is clicked, each field’s name and value are sent to the web server.

• Script form.php then accesses the value for each specific field through the global array $POST.
How the Form Example Works (cont.)

- The superglobal arrays are associative arrays predefined by PHP that hold variable acquired from the user input, the environment, or the web server and are accessible in any variable scope.
  - If the information from the form had been submitted via the HTTP method `get`, then the superglobal array `$_GET` would contain the name-value pairs.

- Since the HTML form and the PHP script “communicate” via the name-value pairs, it is a good idea to make the XHTML object names meaningful so that the PHP script that retrieves the data is easier to understand.
RegisterGlobals

- In PHP versions 4.2 and higher, the directive `register_globals` is set to Off by default for security reasons.

- Turning off `register_globals` means that all variables sent from an XHTML form to a PHP document now must be accessed using the appropriate superglobal array (either `$_POST` or `$_GET`).

- When this directive was turned On, as was the default case in PHP versions prior to 4.2, PHP created an individual global variable corresponding to each form field.
Validation of Form Generated Data

• The form example illustrates an important concept in the validation of user input. In this case, we simply checked the validity of the format of the telephone number entered by the client user.

• In general, it is crucial to validate information that will be entered into database or used in mailing lists. For example, validation can be used to ensure that credit-card numbers contain the proper number of digits before the numbers are encrypted to a merchant.

• In this case, the form.php script is implementing the business logic or business rules for our application.
Pattern Matching in PHP

• For powerful string comparisons (pattern matching), PHP provides functions `ereg` and `preg_match`, which use regular expressions to search a string for a specified pattern.

• Function `ereg` uses Portable Operating System Interface (POSIX) extended regular expressions.
  – POSIX-extended regular expressions are a standard to which PHP regular expression conform.

• Function `preg_match` provides Perl-compatible regular expressions.

• Perl-compatible regular expressions are more widely used than POSIX regular expressions. PHP’s support for Perl-compatible regular expressions eases migration from Perl to PHP. The following examples illustrates these concepts.
expression.php - Example

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- expression.php -->
<!-- Using regular expressions -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Regular expressions</title>
</head>
<body>
<?php
$search = "Now is the time";
print("Test string is: \"$search\"\n\n" );
// call function ereg to search for pattern 'Now' in variable search
if ( ereg( "Now", $search ) )
print("String 'Now' was found.<br/>");

// search for pattern 'Now' in the beginning of the string
if ( ereg( "^Now", $search ) )
print("String 'Now' found at beginning of the line.<br/>");

// search for pattern 'Now' at the end of the string
if ( ereg( "Now$", $search ) )
print("String 'Now' was found at the end of the line.<br/>");

^ matches at beginning of a string

$ matches at end of a string
Uses a regular expression to match a word ending in “ow”.

```php
// search for any word ending in 'ow'
if ( ereg( "\[(\[:<:]\)([a-zA-Z]*ow)\]\[:>:\]", $search, $match ) )
    print( "Word found ending in 'ow': " .
            $match[ 1 ] . "<br />");

// search for any words beginning with 't'
print( "Words beginning with 't' found: ");

while ( eregi( "\[(\[:<:]\)(t[\[:alpha:\]+])\]\[:>:\]",
                   $search, $match ) ) {
    print( $match[ 1 ] . " " );

    // remove the first occurrence of a word beginning
    // with 't' to find other instances in the string
    $search = ereg_replace( $match[ 1 ], "", $search );
}

    print( "<br />");
?>
</body>
</html>
```
Output From `expression.php` - Example

Test string is: 'Now is the time'

String 'Now' was found.
String 'Now' found at beginning of the line.
Word found ending in 'ow': Now
Words beginning with 't' found: the time
Verifying a Username and Password Using PHP

- It is often the case that a private website is created which is accessible only to certain individuals.

- Implementing privacy generally involves username and password verification.

- In the next example, we’ll see an XHTML form that queries a user for a username and password. The fields USERNAME and PASSWORD are posted to the PHP script verify.php for verification.

  - For simplicity, data is not encrypted before sending it to the server.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- password.html -->
<!-- XHTML form sent to password.php for verification -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Verifying a username and a password.</title>
<style type = "text/css">
  td { background-color: #DDDDDD }
</style>
</head>
<body style = "font-family: arial">
<p style = "font-size: 18pt">
<font color=red><B>Welcome to the CNT 4714 High Security WebPage</B></font><HR>
<p style = "font-size: 13pt">
  Type in your username and password below.<br />
  Note that password will be sent as plain text - encryption not used in this application
</p>
</p>
</body>
</html>
<!-- post form data to password.php -->
<form action = "password.php" method = "post">
<br />
<table border = "3" cellspacing = "3" style = "height: 90px; width: 150px; font-size: 10pt" cellspacing = "1">
<tr>
<td colspan = "3">  <strong>Username:</strong>  </td>
</tr>
<tr>
<td colspan = "3"> <input size = "40" name = "USERNAME" style = "height: 22px; width: 115px" />
</td>
</tr>
<tr>
<td colspan = "3">  <strong>Password:</strong>    </td>
</tr>
<tr>
<td colspan = "3">  <input size = "40" name = "PASSWORD" style = "height: 22px; width: 115px" type = "password" />
</td>
</tr>
<tr>
<td colspan = "1">
<input type = "submit" name = "Enter" value = "Enter" style = "height: 23px; width: 47px" />
<td colspan = "2">  <input type = "submit" name = "NewUser" value = "New User" style = "height: 23px" />
</td>
</tr>
</table>
</form>
<HR>
</body>
</html>
<?php
extract( $_POST );
// check if user has left USERNAME or PASSWORD field blank
if ( !$USERNAME || !$PASSWORD ) {
    fieldsBlank();
die();
}
// check if the New User button was clicked
if ( isset( $NewUser ) ) {
    // open password.txt for writing using append mode
    if ( !($file = fopen( "password.txt", "a" ) ) ) {
        // print error message and terminate script
        // execution if file cannot be opened
        print( "<title>Error</title></head><body>
            Could not open password file
        </body></html>" );
die();
}
// write username and password to file and call function userAdded
fputs($file, "$USERNAME,$PASSWORD\n" );
userAdded($USERNAME );
} 
else {

// if a new user is not being added, open file
// for reading
if ( !( $file = fopen( "password.txt", "r" ) ) ) {
    print("<title>Error</title></head>
        <body>Could not open password file
        </body></html>" );
    die();
}

$userVerified = 0;

// read each line in file and check username and password
while ( !feof($file) &amp;&amp; !$userVerified ) {

// read line from file
$line = fgets($file, 255 );

// remove newline character from end of line
$line = chop($line );

// split username and password using comma delimited string
$field = split( ",", $line, 2 );
// verify username
if ( $USERNAME == $field[0] ) {
    $userVerified = 1;

    // call function checkPassword to verify user’s password
    if ( checkPassword( $PASSWORD, $field ) == true )
        accessGranted( $USERNAME );
    else
        wrongPassword();
}

// close text file
fclose( $file );

// call function accessDenied if username has not been verified
if ( !$userVerified )
    accessDenied();

// verify user password and return a boolean
function checkPassword( $userpassword, $filedata )
{
    if ( $userpassword == $filedata[1] )
        return true;
    else
        return false;
}
function userAdded( $name ) {
    print( "<title>Thank You</title></head>
    <body style = "font-family: arial; font-size: 1em; color: blue">
    <strong>You have been added
to the user list, $name. Please remember your password.
    <br />Enjoy the site.</strong>" );
}

function accessGranted( $name ) {
    print( "<title>Thank You</title></head>
    <body style = "font-family: arial; font-size: 1em; color: blue">
    <strong>Permission has been
    granted, $name. <br />
    Enjoy the site.</strong>" );
}

function wrongPassword() {
    print( "<title>Access Denied</title></head>
    <body style = "font-family: arial; font-size: 1em; color: red">
    <strong>You entered an invalid
    password.<br />Access has
    been denied.</strong>" );
}
// print a message indicating access has been denied
function accessDenied() {
    print( "<title>Access Denied</title></head>
    <body style = "font-family: arial;
    font-size: 1em; color: red">
    <strong>
    You were denied access to this server.
    <br/>
    </strong>
    );
}

// print a message indicating that fields
// have been left blank
function fieldsBlank() {
    print( "<title>Access Denied</title></head>
    <body style = "font-family: arial;
    font-size: 1em; color: red">
    <strong>
    Please fill in all form fields.
    <br/>
    </strong>
    );
}
?>
</body>
</html>
Welcome to the CNT 4714 High Security WebPage

Type in your username and password below.

Note that password will be sent as plain text - encryption not used in this application

Execution of password.html. Client-side XHTML form. User clicks on New User button to enter their information.

You have been added to the user list, Mark Llewellyn. Please remember your password. Enjoy the site.

Execution of password.php to enter a new user.
Execution of password.html. Client-side XHTML form. User clicks on Enter button to submit and verify their information.

Execution of password.php to invalidate an attempted entry by a user.

You entered an invalid password. Access has been denied.
How password.php Works

- The PHP script `password.php` verifies the client’s username and password by querying a database. For this example, the “database” of usernames and passwords is just a text file (for simplicity). Existing users are validated against this file, and new users are appended to it.

- Whether we are dealing with a new user is determined by calling function `isset` to test if variable `$NewUser` has been set.

- When the user submits the password.html form to the server, they click either Enter or New User button. After calling function `extract`, either variable `$NewUser` or `$Enter` is created depending on which button was selected. If `$NewUser` has not been set, we assume the user clicked Enter.