



Emailing with PHP

Webworks – A Workshop Series in Web Design (Session 7)

Table of Contents:

1. Forms: A Quick Review
2. PHP's mail() function
3. Making an E-card
4. Appendix

1. Forms: A Quick Review

A Simple Form

To illustrate basic form requirements and syntax, look at the following example:

```
<form action="formprocess.php" method="post">
  Name: <input type="text" name="username"><br>
  Password: <input type="password" name="password"><br>
  Gender: <input type="radio" name="male" checked> Male
         <input type="radio" name="female"> Female <br>
  What would you rather be doing right now?<br>
  <textarea name="info"></textarea><br>
  <input type="submit" name="submit" value="Submit">
  <input type="reset" value="reset">
</form>
```

The code produces this result in the browser:

Name: Max
Password: *****
Gender: Male Female
What would you rather be doing right now?
lots of stuff
Submit reset

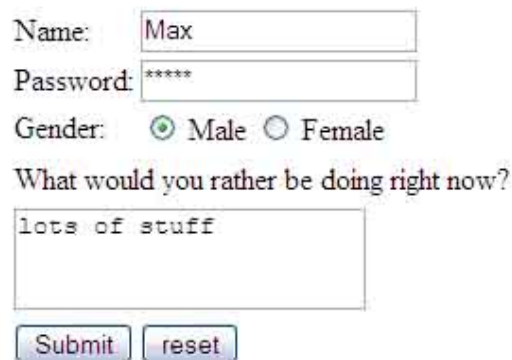
You should note these things in particular:

- The `action` attribute in the `<form>` tag simply tells the browser where the form will be processed. This could even be the same page as the form itself.

- Pay close attention to what you name the input fields. These will eventually become variables in your PHP form. Input names are case sensitive.
- The `value` attribute in the `submit` and `reset` fields is simply what you want to appear in the submit and reset buttons.

Making Your Form Look Nice

Although a cluttered form is still perfectly functional, you can make your forms look more professional by using tables and adjusting the lengths of the fields. Formatting the above form was a bit tricky because of the text area field, but the result is shown below. The revised code is appended to the back of this worksheet.



2. PHP's `mail()` function

The basic function has this format. With the proper use of variables, you can make this all fit on one line:

```
$email = "youremail@stanford.edu";  
$subject = "This is the subject";  
$message = "<i>This is the message</i>";  
  
mail($email, $subject, $message);
```

Sending Your First Email

Go to <http://www.harryzhong.com/webworks/>. Enter in a username, and once you get to the coding page, copy the code on the screen to Notepad. You'll need it later.

Using the format above, send yourself a message. Now check your email.

There are two issues we have to address before we go any further:

- The "From" field says "Nobody [nobody@alexandria23.alexsrv23.com]." As you may have noticed, the mail function doesn't have a "From" parameter. The default sender depends on how the PHP server is configured.

- o You naturally would expect to see *"This is the message"* in italics like you wanted, but this is what you get: "`<i>This is the message</i>`". This is because the default setting for email clients is to format messages as plain text.

Luckily, these two problems are easy to fix with the mail functions 4th, optional parameter: headers!

Using Headers

Headers contain information about the email type and who sent it, but you typically don't see them unless you reply to or forward an email. Headers usually look like this:

```
From: Breanna Chia [mailto:breanna@stanford.edu]
Sent: Wednesday, February 09, 2005 12:52 PM
To: Max Garcia
Subject: Re: hey
```

For the purposes of this discussion, we'll only be using the two headers that address the problems above. If you want the email you send to display your own email address, and activate HTML in the email, simply include in the header:

```
$header = $headers = "MIME-Version: 1.0\r\n";
$header .= "Content-type: text/html; charset=iso-8859-1\r\n";
$header .= "Content-Transfer-Encoding: 7bit\r\n";
$header .= "From: " . $sender_email . " \r\n";
```

The first three lines of the headers tell your email client to expect HTML. As you can recall, you append to a variable using the `.=` operator. The `\r\n` characters create new lines in the headers. Now go ahead and resend that email you had from above. The message should be formatted the way you want it.

On a side note, you may have realized that you don't have to necessarily put *your own* email address as the sender. Now you have a basic idea of how spammers operate.

3. Making an E-Card

This e-card was coded with the purpose of fitting all in one page. The following pseudocode gives the general idea behind it:

```
if (the form has not been submitted) {
    Display a form for the user to fill out;
}
else {
    Format and store the form fields in variables;
    Send the email using those variables;
    if (the email has been sent successfully) {
        Display what has been sent;
    }
    else {
        Display an error message;
    }
}
```

```
}

```

To use this code, you need to change the `form` tag so that the action is `yourname2.php`. The tag in its entirety is `<form action="yourname2.php" method="post">`. This tells the browser that the code needed to process the form is located on the same page as the form itself. Because we're short on time, let us now focus on the `else` statement of the code.

First, locate all the `<input>` fields in the form and note their names. You'll need a variable for each of these except "submit."

```
$name = $_POST['name'];
$email = $_POST['email'];
$subject = $_POST['subject'];
$message = $_POST['message'];
$sender_name = $_POST['sender_name'];
$sender_email = $_POST['sender_email'];

```

Recall that the `$_POST` array simply transfers data from one page to the next. You can access any value in the array (designated by `<input name="value">`) with the call `$_POST['value']`.

To send the email and make a confirmation page, you need to insert the variables in the proper places. Don't forget to put the sender email variable in the From header. You can easily check if your email has been sent because `mail()` returns true if it was successful. And since you already declared the variables, it is no problem to list them again on the confirmation page.

Here's the rest of the code to make the page complete:

```
$mailsent = mail($email, $subject, $message, $header);
if ($mailsent) {
?>
    <html>
        <head>
            <title>Success!
            </title>
        </head>
        <body>
            Thank you, <?php echo $sender_name ?>. Your card has been sent
            successfully to <?php echo $name ?> [<?php echo $email ?>]. <br>
            They have received this message:<br>
            <b>From:</b> <?php echo $sender_email ?><br>
            <b>Subject:</b> <?php echo $subject ?><br>
            <b>Message:</b><br>
            <?php echo $message ?>
        </body>
    </html>
<?php
}
else {
    echo "This page has encountered an error. Please try again.";
}

```

```
}
```

To make things easier, the complete version of the code is available at <http://www.stanford.edu/~maxican/webworks/email.txt>. Remember, the only thing you need to change to make it work is `<form action="yourname2.php">`.

4. Appendix

Formatting Your Form

This is the code I used to make the example form look nicer. There are many ways you can go about doing this.

```
<form action="formprocess.php" method="post">
<table>
  <tr>
    <td>Name:
    </td>
    <td><input type="text" name="username">
    </td>
  </tr>
  <tr>
    <td>Password:
    </td>
    <td><input type="password" name="password">
    </td>
  </tr>
  <tr>
    <td>Gender:
    </td>
    <td><input type="radio" name="male" checked> Male
      <input type="radio" name="female"> Female
    </td>
  </tr>
</table>
<table>
  <tr>
    <td>What would you rather be doing right now?
    </td>
    <td>&nbsp;
    </td>
  </tr>
  <tr>
    <td colspan=2><textarea name="info" cols=24></textarea>
    </td>
    <td>&nbsp;
    </td>
  </tr>
  <tr>
    <td colspan=2>
      <input type="submit" name="submit" value="Submit">
      <input type="reset" value="reset">
    </td>
  </tr>
</table>
</form>
```