PHP Form Handling

The PHP super globals $_GET and $_POST are used to collect form-data.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. It is an associative array of variables passed to the current script via the URL parameters (aka. query string). Note that the array is not only populated for GET requests, but rather for all requests with a query string. You can send limited amount of data through get request. The GET variables are passed through urldecode().In general, a URL with GET data will look like this:

http://www.example.com/action.php?name=xyz&age=32

$_GET examples:

```
<?php
echo 'Hello ' . htmlspecialchars($_GET["name"]) . '!';
?>
Assuming the user entered http://example.com/?name=Hannes
```

The above example will output something similar to:

Hello Hannes!

File: form1.html

```
1. <form action="welcome.php" method="get">
2. Name: <input type="text" name="name"/>
3. <input type="submit" value="visit"/>
4. </form>
```

File: welcome.php

```
1. <?php
2. $name=$_GET["name"];//receiving name field value in $name variable
3. echo "Welcome, $name";
4. ?>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP GET method.

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc. The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of

data through post request. It is an associative array of variables passed to the current script via the HTTP POST method when using application/x-www-form-urlencoded or multipart/form-data as the HTTP Content-Type in the request.

$_POST examples:

```
<?php
echo 'Hello ' . htmlspecialchars($_POST["name"]) . '!';
?>
```

Assuming the user POSTed name=Hannes

The above example will output something similar to:

Hello Hannes!

File: form1.html

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```

File: login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password variable
echo "Welcome: $name, your password is: $password";
?>
```

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are super globals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases. GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Basic instructions for setting up a form handler in PHP to verify user input and send an email or display an error message in case the validation fails:

1. Sample HTML Form

Here is the HTML and PHP code for the form we will be working with:

```
<form method="POST" action="<?PHP echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" accept-charset="UTF-8">
<p><label>Your Name<strong>*</strong><br>
<input type="text" size="48" name="name" value="<?PHP
if(isset($_POST['name'])) echo htmlspecialchars($_POST['name']);
?>"></label></p>
<p><label>Email Address<strong>*</strong><br>
<input type="email" size="48" name="email" value="<?PHP
if(isset($_POST['email'])) echo htmlspecialchars($_POST['email']);
?>"></label></p>
<p><label>Subject<br>
<input type="text" size="48" name="subject" value="<?PHP
if(isset($_POST['subject'])) echo htmlspecialchars($_POST['subject']);
?>"></label></p>
<p><label>Enquiry<strong>*</strong><br>
<textarea name="message" cols="48" rows="8"><?PHP
if(isset($_POST['message'])) echo htmlspecialchars($_POST['message']);
?></textarea></label></p>
<p><input type="submit" name="sendfeedback" value="Send Message"></p>

</form>
```

The form will look something like the following:

Your Name*

Email Address*

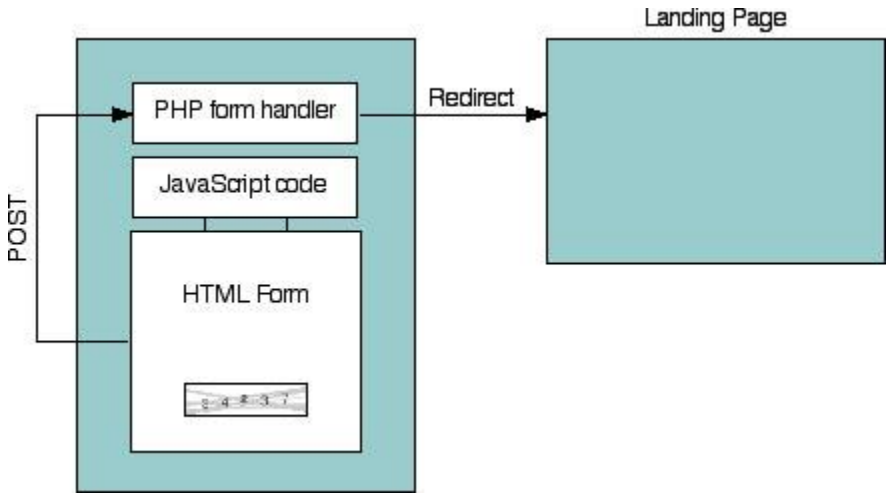
Subject

Enquiry*

Send Message

For testing purposes removed all the usual JavaScript Form Validation and HTML5 Form Validation so the form can simply be submitted and validated by PHP on the server.

PHP is used to insert the form `action` as the current page. That's because we are using the "redirect-after-POST" technique as illustrated here:



It prevents the form from being resubmitted if the landing page is reloaded, and allows us to display validation error messages inline using PHP. Finally, the code includes PHP commands to re-insert any submitted values back in to the form so they don't have to be retyped in case of an error.

2. PHP Form Handler

The important characteristics of a form handler is that it verifies that the required variables have been set, and that they have appropriate values. Remember to be thorough as this is your last (only real) line of defence against malicious scripts. Here we are detecting a `POST` event and extracting the values directly from the PHP `$_POST` array for testing:

```
<?PHP
  // form handler
  if($_POST && isset($_POST['sendfeedback'], $_POST['name'],
$_POST['email'], $_POST['subject'], $_POST['message'])) {

    $name = $_POST['name'];
    $email = $_POST['email'];
    $subject = $_POST['subject'];
    $message = $_POST['message'];

    if(!$name) {
      $errorMsg = "Please enter your Name";
    } elseif(!$email || !preg_match("/^\S+@\S+$/", $email)) {
      $errorMsg = "Please enter a valid Email address";
    } elseif(!$message) {
      $errorMsg = "Please enter your comment in the Message box";
    } else {
      // send email and redirect
      $to = "feedback@example.com";
      if(!$subject) $subject = "Contact from website";
      $headers = "From: webmaster@example.com" . "\r\n";
      mail($to, $subject, $message, $headers);
      header("Location: http://www.example.com/thankyou.html");
      exit;
    }

  }

?>
```

All HTML form elements, aside from unselected checkboxes and radio buttons, will appear in the `$_POST` array, even if the value is blank. This includes the `submit` button, which in this case we have named `sendfeedback`. Naming the button is useful in case there are multiple forms on the page. The first thing the form handler does is check that all the fields in our form, including the button, appear in the POST array. Then we extract the relevant values for testing.

The testing here is fairly rudimentary. In reality we have special functions for validating email addresses and other data types - as will most JavaScript libraries. We also have more advanced functions for sending email. For public-facing forms

you should add a CAPTCHA or similar device, as you can see in our Feedback form below, or risk being bombarded by spambots.

3. Placement of the code

The PHP code needs to appear at the top of the page - before any HTML or whitespace is displayed. Otherwise the redirect will fail with the ubiquitous warning "Cannot modify header information - headers already sent". Your final code should look something like this:

```
<?PHP
  // php form handler and redirect
?>
<!DOCTYPE html>
<html>

<head>
  <title>...</title>
  ...
</head>

<body>

<form method="POST" action="..." ...>
<!-- html form -->
</form>

<script type="text/javascript">
  // javascript form validation
</script>

</body>

</html>
```

It doesn't actually matter where on the page the JavaScript appears, whether inline or as a link.

4. Displaying Error Messages

When the form is submitted, but not validated, the code execution flows through to the page. All we need to do is check for an error message and display it on the page:

```
<?PHP
```

```
  if(isset($errorMsg) && $errorMsg) {
    echo "<p style=\"color:
red;\">*",htmlspecialchars($errorMsg),"</p>\n\n";
  }
?>
```

Again, a more advanced version would place the error message next to the affected field, and do this for multiple fields at once. In the demonstration above we've included the error message at the top of the form.

5. Avoiding Global Variables

In the previous example we made a *faux pas* in polluting the global variable space. We can avoid this, and make our code more modular and reusable, by calling it as a function:

```
<?PHP
  // form handler
  function validateFeedbackForm($arr)
  {
    extract($arr);

    if(!isset($name, $email, $subject, $message)) return;

    if(!$name) {
      return "Please enter your Name";
    }
    if(!preg_match("/^\S+@\S+$/", $email)) {
      return "Please enter a valid Email address";
    }
    if(!$subject) $subject = "Contact from website";
    if(!$message) {
      return "Please enter your comment in the Message box";
    }

    // send email and redirect
    $to = "feedback@example.com";
    $headers = "From: webmaster@example.com" . "\r\n";
    mail($to, $subject, $message, $headers);
    header("Location: http://www.example.com/thankyou.html");
    exit;
  }

  // execution starts here
  if(isset($_POST['sendfeedback'])) {
    // call form handler
    $errorMsg = validateFeedbackForm($_POST);
  }

?>
```

The extract method turns key/value pairs from the $_POST array into separate variables, but only for the scope of the function. We could go further and create a form validation class, with separate methods for validating text, email, dates, etc, but that's a project in itself.

The remaining code on the page stays the same:

```
<form method="POST" action="<?PHP echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" accept-charset="UTF-8">
<?PHP
  if(isset($errorMsg) && $errorMsg) {
    echo "<p style=\"color:
red;\">*",htmlspecialchars($errorMsg),"</p>\n\n";
  }
?>
<p><label>Your Name<strong>*</strong><br>
<input type="text" size="48" name="name" value="<?PHP
if(isset($_POST['name'])) echo htmlspecialchars($_POST['name']);
?>"></label></p>
<p><label>Email Address<strong>*</strong><br>
<input type="email" size="48" name="email" value="<?PHP
if(isset($_POST['email'])) echo htmlspecialchars($_POST['email']);
?>"></label></p>
<p><label>Subject<br>
<input type="text" size="48" name="subject" value="<?PHP
if(isset($_POST['subject'])) echo htmlspecialchars($_POST['subject']);
?>"></label></p>
<p><label>Enquiry<strong>*</strong><br>
<textarea name="message" cols="48" rows="8"><?PHP
if(isset($_POST['message'])) echo htmlspecialchars($_POST['message']);
?></textarea></label></p>
<p><input type="submit" name="sendfeedback" value="Send Message"></p>
</form>
```

PHP - Dealing with Multi-Value Fields:

The following form fields are capable of sending multiple values to the server:

<label **for**=**"favoriteWidgets"** >What are your favorite widgets?</label>

<select name=**"favoriteWidgets"** id=**"favoriteWidgets"** size=**"3"** multiple=**"multiple"** >

<option value=**"superWidget"** >The SuperWidget</option>

<option value=**"megaWidget"** >The MegaWidget</option>

<option value=**"wonderWidget"** >The WonderWidget</option>

</select>

<label **for**=**"newsletterWidgetTimes"** >Do you want to receive our **'PhoneTimes'**newsletter?</label>

<input type=**"checkbox"** name=**"newsletter"** id=**"newsletterWidgetTimes "** value=**"widgetTimes"** />

<label **for**=**"newsletterFunWithWidgets"** >Do you want to receive our **'Fun with Widgets'**newsletter?</label>

<input type=**"checkbox"** name=**"newsletter"** id=**"newsletterFunWithWidgets"** value=**"funWithWidgets"** />

The first form field is a multi-select list box, allowing the user to pick one or more (or no) options. The second two form fields are checkboxes with the same name (newsletter) but different values (widgetTimes and funWithWidgets). If the user checks both checkboxes then both values, widgetTimes and funWithWidgets, are sent to the server under the newsletter field name.

To handle multi-value fields in your PHP scripts, add square brackets ([]) after the field name in your HTML form. Then, when the PHP engine sees a submitted form field name with square brackets at the end, it creates a nested array of values within the $_GET or $_POST (and $_REQUEST) super global array, rather than a single value. You can then pull the individual values out of that nested array.

So you might create a multi-select list control as follows:

<select name=**"favoriteWidgets[]"** id=**"favoriteWidgets"** size=**"3"** multiple=**"multiple"** ... < /select >

You'd then retrieve the array containing the submitted field values as follows:

$favoriteWidgetValuesArray = $_GET[**" favoriteWidgets"**]; **// If using get method**

$favoriteWidgetValuesArray = $_POST[**" favoriteWidgets"**]; **// If using post method**

Example:

The form handler deals with these multi-value fields, displaying their values within the Web page.

<html>
<head>
<title>Membership Form</title>
</head>
<body>
Membership Form
Thanks **for** choosing to join The PhoneClub. To register, please fill in your details below **and** click Send Details.

```
<form action="process_registration_multi.php" method="post" >
  <div style="width: 30em;" >
    <label for="firstName" >First name</label>
    <input type="text" name="firstName" id="firstName" value="" />

    <label for="lastName" >Last name</label>
    <input type="text" name="lastName" id="lastName" value="" />

    <label for="password1" >Choose a password</label>
    <input type="password" name="password1" id="password1" value="" />
    <label for="password2" >Retype password</label>
    <input type="password" name="password2" id="password2" value="" />

    <label for="genderMale" >Are you male...</label>
    <input type="radio" name="gender" id="genderMale" value="M" />
    <label for="genderFemale" >...or female?</label>
    <input type="radio" name="gender" id="genderFemale" value="F" />

    <label for="favoriteWidgets" >What are your favorite widgets?</label>
    <select name="favoriteWidgets[]" id="favoriteWidgets" size="3"
multiple="multiple" >
      <option value="superWidget" >The SuperWidget</option>
      <option value="megaWidget" >The MegaWidget</option>
      <option value="wonderWidget" >The WonderWidget</option>
    </select>

    <label for="newsletterWidgetTimes" >Do you want to receive our
'PhoneTimes' newsletter?</label>
    <input type="checkbox" name="newsletter[]"
id="newsletterPhoneTimes" value="widgetTimes" />
    <label for="newsletterFunWithWidgets" >Do you want to receive our 'Fun
with Widgets'newsletter?</label>
    <input type="checkbox" name="newsletter[]" id="newsletterFunWith
Widgets" value="funWithWidgets" />
    <label for="comments" >Any comments?</label>
    <textarea name="comments" id="comments" rows="4" cols="50"
></textarea>

    <div style="clear: both;" >
      <input type="submit" name="submitButton" id="submitButton"
value="Send Details" />
```

```
            <input type="reset" name="resetButton" id="resetButton" value="Reset
Form" style="margin-right: 20px;" />
          </div>
        </div>
      </form>

    </body>
  </html>
```

Now save the following script as process_registration_multi.php in your document root folder:

```
  <html>
    <head>
      <title>Thank You</title>
    </head>
    <body>
      Thank You
      <p>Thank you for registering. Here is the information you submitted:</p>
  <?php

  $favoriteWidgets ="" ;
  $newsletters ="" ;

  if (isset($_POST[" favoriteWidgets" ])) {
    foreach ($_POST[" favoriteWidgets" ] as $widget) {
      $favoriteWidgets.= $widget." ," ;
    }
  }

  if (isset($_POST[" newsletter" ])) {
    foreach ($_POST[" newsletter" ] as $newsletter) {
      $newsletters.= $newsletter." ," ;
    }
  }
 $favoriteWidgets = preg_replace(" /, $/" ," " , $favoriteWidgets);
 $newsletters = preg_replace(" /, $/" ," " , $newsletters);
?>
    <dl>
      <dt>First name</dt><dd><?php echo $_POST[" firstName" ]?></dd>
      <dt>Last name</dt><dd><?php echo $_POST[" lastName" ]?></dd>
      <dt>Password</dt><dd><?php echo $_POST[" password1" ]?></dd>
      <dt>Retyped password</dt><dd><?php echo $_POST[" password2" ]?></dd>
```

```
        <dt>Gender</dt><dd><?php echo $_POST[" gender" ]?></dd>
        <dt>Favorite widgets</dt><dd><?php echo $favoriteWidgets?></dd>
        <dt>You want to receive the following newsletters:</dt><dd>
        <?php echo $newsletters?></dd>
        <dt>Comments</dt><dd><?php echo $_POST[" comments" ]?></dd>
      </dl>
    </body>
  </html>
```

References:

https://www.javatpoint.com/php-form

https://www.w3schools.com/php/php_forms.asp

https://www.php.net/manual/en/reserved.variables.php

https://www.the-art-of-web.com/php/form-handler/

https://www.tutorialspoint.com/php/php_form_introduction.htm

http://www.java2s.com/example/php-book/dealing-with-multi-value-fields.html