Episode 6

PHP Forms

Introduction

Welcome back to this episode of PHP Fundamentals. PHP applications are generally used to transmit data from the user to the server and back. In this section, we are going to learn how to use forms to send data to the server and get data back. We are also going to take our first step in form security.
Let's get started.

Goals

In this section of the course your goals are:

- [ ] To handle a form using GET.
- [ ] To handle a form using POST.
- [ ] To learn secure ways to send and receive data using htmlspecialchars( )

GET and POST

Let's take a look at an HTML file that contains a form:

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Forms</title>
    <link type="text/css" href="forms.css" rel="stylesheet">
</head>
<body>
    <main>
    <h2>Rubik's Cube Time</h2>
    <form action="forms.php" method="get">
        <label>Name: </label>
        <input type="text" name="firstName"/><br>
        <label>Time: </label>
        <input type="text" name="time"/><br>
        <input type="submit" value="Enter">
    </form>
        </main>
</body>
</html>
```

This is the CSS file:

```css
body {
    margin: 30px;
}
main {
    width: 20rem;
    margin: 0 auto;
    border: solid 1px navy;
    padding: 1em;
}
h2 {
    text-align: center;
    color: navy;
    margin-top: 0;
}
label {
    width: 10rem;
    float: left;
    margin-bottom: 1em;
}
input {
    width: 9.625rem;
    margin-bottom: 1em;
    padding-bottom: 0;
    text-align: center;
}
input[type="submit"] {
    margin-left: 3rem;
    margin-right: 3rem;
    width: 70%;
}
br {
    clear: left;
}
```

This is a simple form with two labels, two input fields and a submit button. When the user enters the data it uses the **get** method and sends the data to the file **forms.php** for processing. Let’s take a look at the php file:

```
<?php
$name = $_POST['firstName'];
$time = $_POST['userTime'];
$myTime = 57.18;
$timeDiff = $myTime - $time;
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Forms</title>
    <link type="text/css" href="forms.css" rel="stylesheet">
</head>
<body>
    <main>
    <?php
    echo "Wow ".$name." ".$time. " seconds!!!<br>";
    echo "You beat me by ".$timeDiff." seconds";
    ?>
    </main>
</body>
</html>
```

This is where the action happens. Because we used the **get** method in the HTML form, we have to use the **$_GET** array in order to retrieve the data. To get the value that the user entered we need to use the value of the **name** attribute of the input field. The two values of the name attribute are **firstName** and **userTime**. We have also created two variables **name** and **time** to store the data entered by the user. Lastly we have created two additional variables to help us send data back to the user.
Now I want you to take a look at something. After the user submits the data using the form, take a look at the URL field. What you see is the data entered by the user. That, of course, can be a big problem if the user is sending sensitive data. Let's take a look at the **post** method to solve that.
We are going to make two changes. The first is in the HTML file, we are going to change the method from get to post. Second, in the PHP file, we are going to use the $_POST array to retrieve the data sent by the user.
Now let's go back to the form, enter some data and submit it. Once the data is received by the server and displayed in the browser, let's take a look at the URL field. This time the data entered by the user is not shown. In general use, the **get** method when you are retrieving data from the server, and use the **post** method when sending data to the server.

htmlspecialchars( )

You can't always trust input entered by the user. Not only will they make mistakes and enter the wrong type, but more worrisome is the fact that there will always be a malicious user who may enter data that will damage your files or expose sensitive data.
One way to avoid this is by using the **htmlspecialchars( )** function. Using this function will protect against

XSS (Cross-side Scripting) attacks.
Let’s take a look at the following PHP file:

```
<?php
$name = $_POST['firstName'];
$time = $_POST['userTime'];
$myTime = 57.18;
$timeDiff = $myTime - $time;
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Special Chars</title>
    <style>body {margin: 30px;}</style>
</head>
<body>
    <?php
    echo "Wow ".$name." ".$time. " seconds!!!<br>";
    echo "My time was ".$myTime." seconds.<br>";
    echo "You beat me by ".$timeDiff." seconds";
    ?>
</body>
</html>
```

This php file receives two variables from the user and sends back four variables. Those variables that were sent by the user need to be checked before being sent back to the browser: $name and $time. What we are going to do is put those variables inside the htmlspecialchars function. And the resulting code looks like this:

```
<?php
$name = $_POST['firstName'];
$time = $_POST['userTime'];
$myTime = 57.18;
$timeDiff = $myTime - $time;
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Special Chars</title>
    <style>body {margin: 30px;}</style>
</head>
<body>
    <?php
    echo "Wow ".htmlspecialchars($name)." ".htmlspecialchars($time). " seconds!!!<br>";
    echo "My time was ".$myTime." seconds.<br>";
    echo "You beat me by ".$timeDiff." seconds";
    ?>
</body>
</html>
```

That’s it. The first step in securing our applications.

Debugging Code

Examine the following code. Find and fix the errors.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Debugging Forms</title>
    <link type="text/css" href="forms.css" rel="stylesheet">
</head>
<body>
    <main>
    <h2>Form with Erros</h2>
    <form action="debugging.php" method="post">
        <label>First Name: </label>
        <input type="text" name="firstname"/><br>
        <label>Last Name: </label>
        <input type="text" name="lasName"/><br>
        <input type="submit" value="Enter">
    </form>
        </main>
</body>
</html>
```

```
<?php
$firstName = $_GET['firstName'];
$lastName = $_GET['lastName'];
$title = "You Fixed the Errors.";
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Forms</title>
    <link type="text/css" href="forms.css" rel="stylesheet">
</head>
<body>
    <main>
    <h2><?php echo title; ?></h2>
    <h3><?php echo $firstName' '$lastName.' is a master debugger.' ?></h3>
    </main>
</body>
</html>
```

Note: Use the same CSS file as above.

Lab Exercises

For the following exercises use the **forms.html** and the **forms.css** files included in this section. You will

need to make small changes depending on the number of labels and input fields required.
You will also need the php starter file shown below.

```
<?php

?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Forms</title>
    <link rel="stylesheet" type="text/css" href="forms.css">
    <style>
        h2 {
            margin-bottom: 0;
        }
    </style>
</head>
<body>
    <main>

    </main>
</body>
</html>
```

For each exercise include any variable whose value was entered by the user, and has to be sent back to the browser, in the **htmlspecialchars( )** function.

1. *Personal digital clock*.

- Modify forms.html to have only one label “Name” and one input of type text. Keep the input of type submit.
- In the upper portion of the php file add the following:
    - `date_default_timezone_set("America/New_York");`
    - `$time = date("D h:i:s a");`
- In the php file create a variable that will hold the value entered by the user.
- Send a message back to the user that includes the name he entered and the time obtained in the `$time` variable. Include this message within the `main` brackets.
  Note: “America/New_York” will return the Eastern time for the US. For a list of other places around the world go to this List of Supported Time Zones.

2. *Converting Celcius to Fahrenheit*. Write a script that obtains a temperature in Celsius and converts it to Fahrenheit. Display the result to the user.
   The formula to convert Celcius to Fahrenheit is:

$$F = \frac{9}{5} * C + 32$$

Where C is degrees in Celcius.

3. *Wind-chill temperature*. When was the last time you left the house dressed for 50-degree weather, but it felt more like 30. The wind has a big impact on how cold it feels. The wind-chill temperature is obtained by the following formula:

$$W = 35.74 + 0.6215t - 35.75v^{0.16} + 0.4275tv^{0.16}$$

   Where t is the temperature outside, v is the wind speed in miles per hour, and W is the wind-chill temperature.
   Write a script that obtains the temperature and wind speed from the user and calculates and displays the wind-chill temperature.