

# Architecting Web Applications using PHP

## Session 6

### Form Handling

# Session Overview

---

In this session, you will be able to:

- Explain how to create a simple HTML Form
- Compare GET versus POST methods and identify their uses
- Explain `$_REQUEST` variable
- Explain different form validations
- Describe the importance of required fields in PHP forms
- Elaborate the process of validating in PHP forms

# Simple HTML Form [1-3]

A form is used to get data from the users and pass it to the Web server for processing. It is created using HTML tags.

Different types of form controls for collecting data include Check boxes, Radio buttons, Submit and Reset buttons, Clickable controls, and so on.

Data can be submitted to the server for processing using the following methods:

**PHP POST  
method**

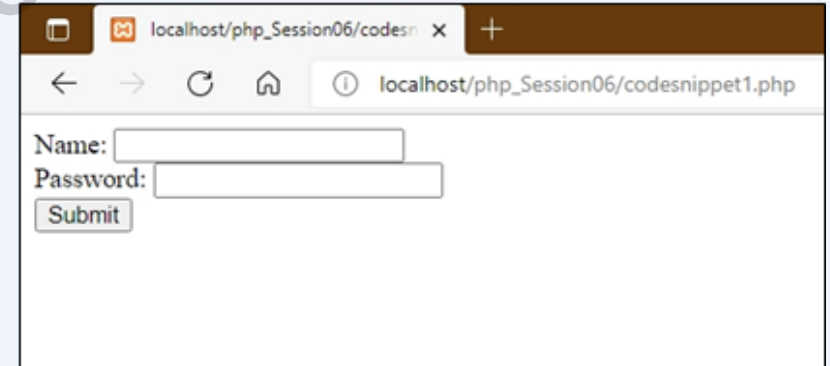
**PHP GET  
method**

# Simple HTML Form [2-3]

## Code Snippet:

```
<html>
  <body>
    <form action="welcome.php" method="post">
      Name: <input type="text" name="name"><br>
      Password: <input type="password"
name="password">
      <br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

## OUTPUT:

A screenshot of a web browser window. The address bar shows the URL 'localhost/php\_Session06/codesnippet1.php'. The page content displays a simple form with two input fields: 'Name:' followed by a text input field, and 'Password:' followed by a password input field. Below the password field is a 'Submit' button.

**Figure: Generating a Simple Form**

Code Snippet shows a simple HTML form with two input fields and a submit button.

# Simple HTML Form [3-3]

## Code Snippet:

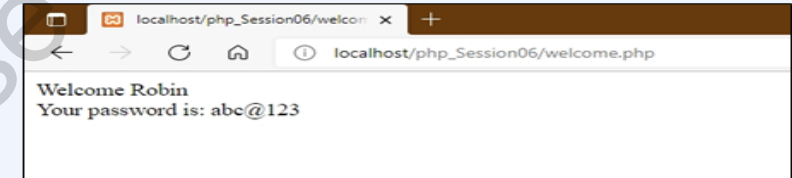
```
Welcome <?php echo $_POST["name"]; ?> <br>
Your password is: <?php echo $_POST["password"]; ?>
```

Code Snippet 2 shows statements to retrieve and display name and password of the user that were submitted through the form.

## Code Snippet:

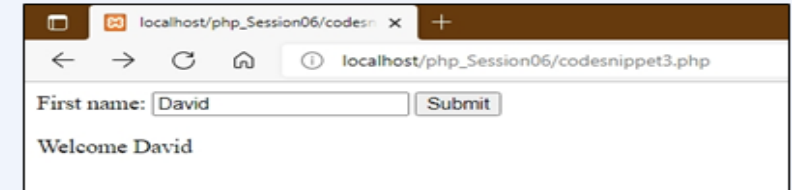
```
<html>
<body>
<form action="codesnippet3.php" method="get">
First name: <input type="text" name="firstname">
<input type="submit" value="Submit">
</form>
Welcome
<?php
// Turn off all error reporting
error_reporting(0);
echo $_GET["firstname"]
?>
</body>
</html>
```

## OUTPUT:



**Figure: Displaying Form Data**

## OUTPUT:



**Figure: Displaying Form Data with GET**

Second Code Snippet demonstrates use of GET to display the name entered by the user with a personalized greeting.

# GET versus POST

Both GET and POST methods each create an array (Example: `array( key1 => value1, key2 => value2, key3 => value3, ...)`).

## Uses of GET

- GET method displays all the values being sent in the URL itself. Hence, this method is recommended only for sending non-sensitive data.
- One should not use the GET method to send passwords or other sensitive information.
- This method also has a limit on the amount of information that can be sent. The limitation is about 2048 characters.
- However, it is possible to bookmark the page.
- That means, a browser can remember the page and it will be easy for you to access it next time.
- Unlike POST method, GET method can be bookmarked because the data required for the request is stored in the URL.

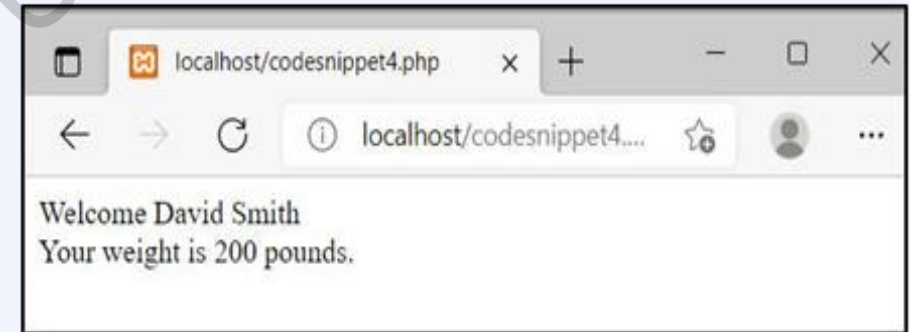
## Uses of POST

- POST method does not display values in the URL because data sent is in the package form and maintained in a separate communication processor.
- It has no limit on the amount of information sent because it is submitted via the HTTP's body.
- Moreover, the POST method supports advanced functionality such as multi-part binary input while uploading files to a server.
- It also supports other data types such as string, numeric, and so on.
- However, as variables are not displayed in the URL, it is not possible to bookmark the page.

# \$\_REQUEST Variable

## Code Snippet:

```
<?php
    error_reporting(0);
    if( $_REQUEST["name"] || $_REQUEST["weight"] )
    {
        echo "Welcome ". $_REQUEST['name']. "<br />";
        echo "You are ". $_REQUEST['weight']. " kgs .";
        exit();
    }
?>
<html>
    <body>
        <form action="<?php $_PHP_SELF ?>"
method="POST">
            Name: <input type="text" name="name" />
            Weight: <input type="text" name="weight" />
            <input type="submit" />
        </form>
    </body>
</html>
```

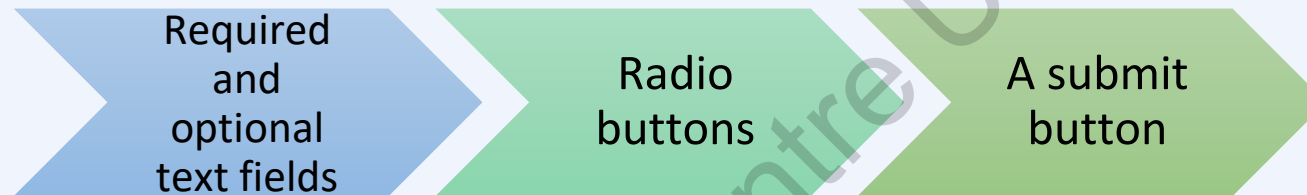


*Figure: Using \$\_REQUEST*

Code Snippet shows the use of \$\_REQUEST variable.

# Form Fields Validation

In general, commonly used input fields in PHP form include:



PHP Form Validation  
\* required field  
UserID:  \*  
Password:  \*  
College:   
Department:   
Comment:   
Gender: ☐ Female ☐ Male ☐ Other \*

**Figure: Form Validation**

Field	Validation Rules	Condition
Name	It must only contain letters and whitespace. The user must provide the input.	Mandatory
e-mail	It must contain a valid e-mail address (with '@' and '.'). The user must provide the input.	Mandatory
Website	If present, it must contain a valid URL.	Optional
Comment	Multi-line input field (textarea). If present user has to write comments in text area given.	Optional
Gender	The user must select one option.	Mandatory

**Table: Form Validation Rules**

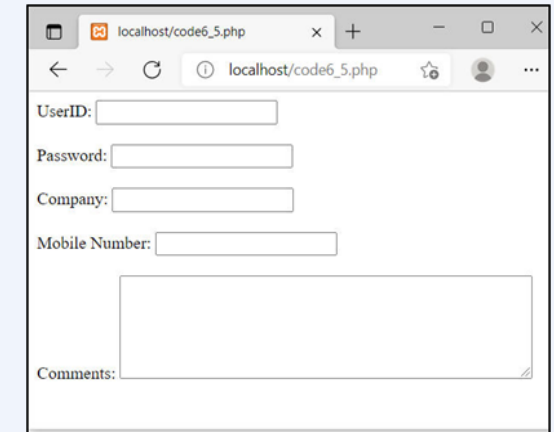


# Text Fields

## Code Snippet:

```
<form action="login.php" method="get">
UserID: <input type="text" name="userID"><br><br>
Password: <input
type="password" name="password"><br><br>
Company: <input type="text" name="company"><br><br>
Mobile Number: <input type="text" name="Mobile
Number"><br><br>
Comments: <textarea name="comments" rows="6" cols="50">
</textarea>
</form>
```

Code Snippet shows how to create a simple form that utilizes text fields for accepting Username, Password, Company, and Comment.

A screenshot of a web browser window displaying a form. The browser's address bar shows 'localhost/code6\_5.php'. The form contains five input fields: 'UserID' (text), 'Password' (password), 'Company' (text), 'Mobile Number' (text), and 'Comments' (a larger text area). Each field is preceded by its label. The form is styled with a light background and simple borders.

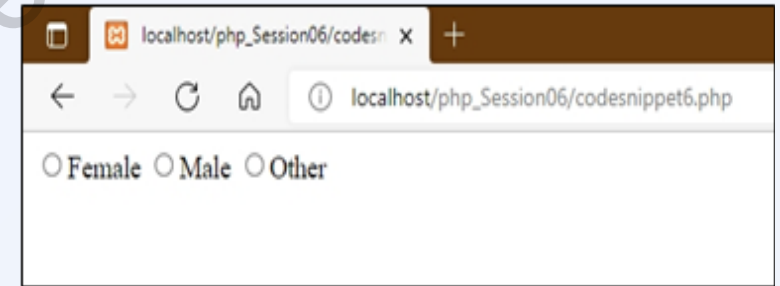
**Figure: Using Text Fields in a Form**

# Radio Buttons

## ***Code Snippet:***

```
<form action="form1.php" method="get">
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
</form>
```

Code Snippet creates gender fields in a form to be selected using radio buttons.



***Figure: Using Radio Buttons in a Form***

# Using `htmlspecialchars()` with Form Elements



Form elements are the elements which are used inside the `form` tag.



In a form, it is recommended that methods such as GET or POST, variables such as `$_SERVER["PHP_SELF"]`, and `htmlspecialchars()` function be used.

For Aptech Centre Use Only

# Validating Form Data with PHP [1-5]

As a first step in validation, one should pass all variables through the PHP's `htmlspecialchars()` function to validate form data.

Consider a scenario where a malicious user tries to submit a hyperlink within a text field that is meant to receive user names or some such text. When the hyperlink is posted to the redirected page, which it can lead to harmful action.

When `htmlspecialchars()` function is used, if a user submits data such as following in a text field, it will not be executed as a hyperlink: `<script> location.href('http://www.attack.com')</script>`.

This is because the `htmlspecialchars()` function caused the data to be saved internally as HTML escaped code: `&lt;script&gt;location.href('http://www.attack.com') &lt;/script&gt;`

Thus, this code can now be displayed safely on a page or within an e-mail because it is treated as text instead of a hyperlink. On the other hand, if `htmlspecialchars()` function had not been used with the text field, the outcome would have been unsafe and potentially dangerous.

# Validating Form Data with PHP [2-5]

## Code Snippet:

```
<!DOCTYPE html>
<head>
</head>
<body>
<?php
// Initialize variables during creation to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = sanitize_data($_POST["name"]);
    $password = sanitize_data($_POST["password"]);
    $company = sanitize_data($_POST["company"]);
    $comment = sanitize_data($_POST["comment"]);
    $gender = sanitize_data($_POST["gender"]);
    echo "Data is sanitized in appropriate format <br>";
}

function sanitize_data($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data, ENT_QUOTES);
    return $data;
}
?>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name">
<br><br>
```

Code Snippet shows a program to check each `$_POST` variable with a `sanitize_data()` function.

# Validating Form Data with PHP [3-5]

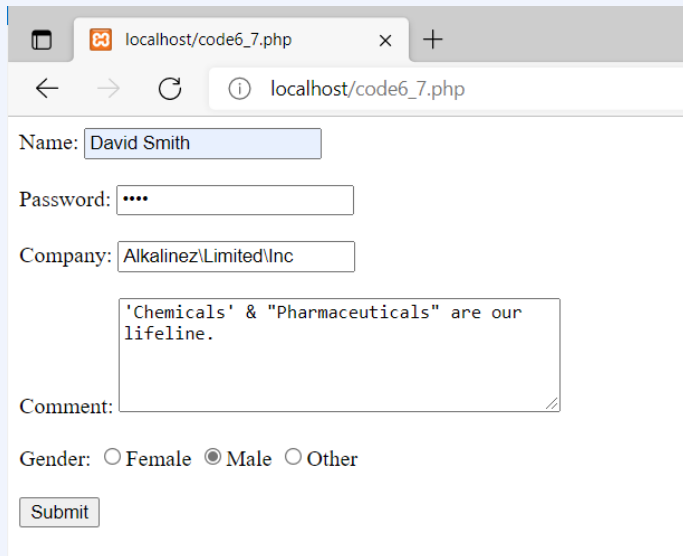
```

Password: <input type="password" name="password">
<br><br>
Company: <input type="text" name="company">
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
<br><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
error_reporting(0);
echo $name;
echo "<br>";
echo $password;
echo "<br>";
echo $company;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>

```

For Aptech Centre Use Only

# Validating Form Data with PHP [4-5]



localhost/code6\_7.php

← → ↻ ⓘ localhost/code6\_7.php

Name:

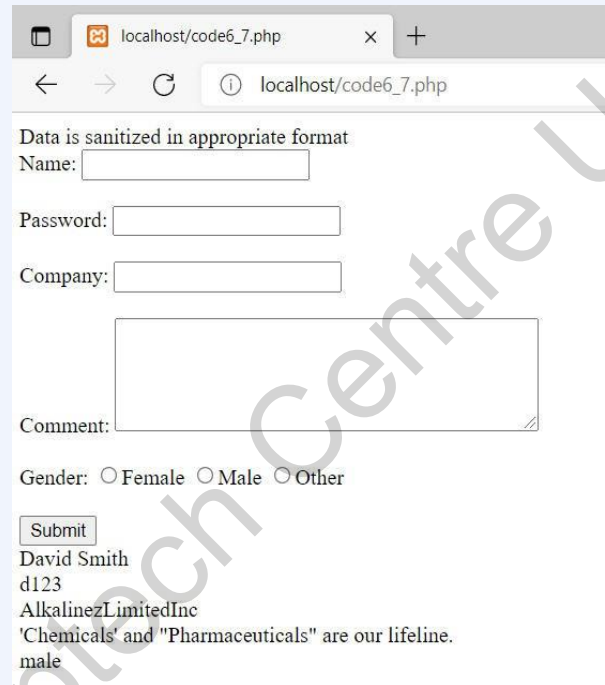
Password:

Company:

Comment:

Gender: ☐ Female ☒ Male ☐ Other

**Figure: Using a Function to Validate Data in a Form**



localhost/code6\_7.php

← → ↻ ⓘ localhost/code6\_7.php

Data is sanitized in appropriate format

Name:

Password:

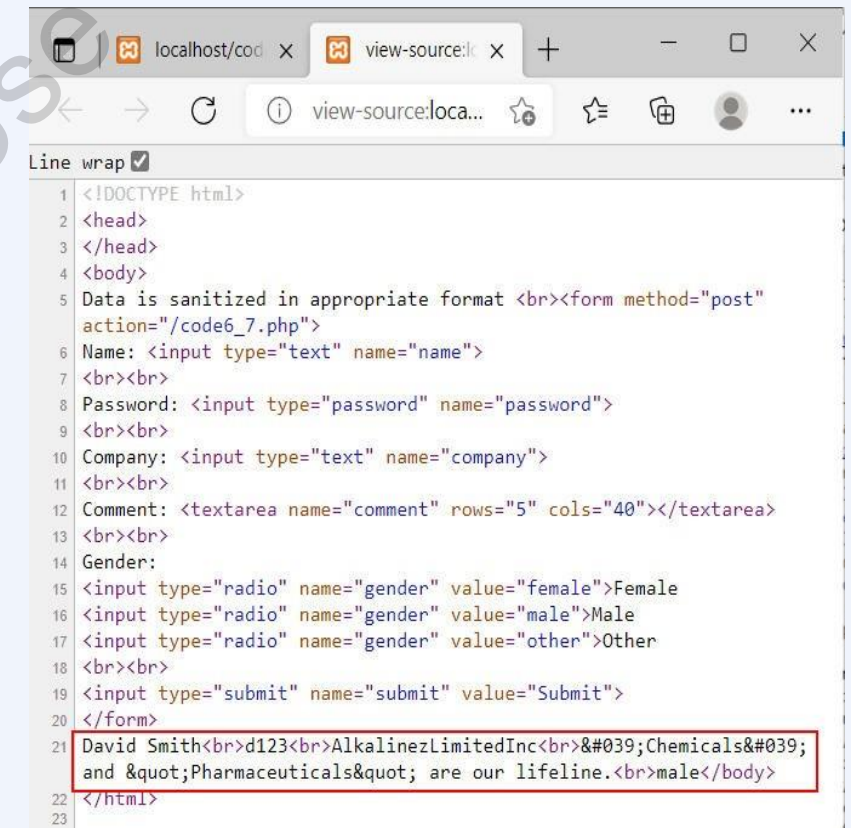
Company:

Comment:

Gender: ☐ Female ☐ Male ☐ Other

David Smith  
d123  
AlkalineZ Limited Inc  
'Chemicals' & 'Pharmaceuticals' are our lifeline.  
male

**Figure: Output After Clicking Submit**



localhost/code6\_7.php view-source:localhost/code6\_7.php

← → ↻ ⓘ view-source:localhost/code6\_7.php

Line wrap ☒

```
1 <!DOCTYPE html>
2 <head>
3 </head>
4 <body>
5 Data is sanitized in appropriate format <br><form method="post"
6 action="/code6_7.php">
7 Name: <input type="text" name="name">
8 <br><br>
9 Password: <input type="password" name="password">
10 <br><br>
11 Company: <input type="text" name="company">
12 <br><br>
13 Comment: <textarea name="comment" rows="5" cols="40"></textarea>
14 <br><br>
15 Gender:
16 <input type="radio" name="gender" value="female">Female
17 <input type="radio" name="gender" value="male">Male
18 <input type="radio" name="gender" value="other">Other
19 <br><br>
20 <input type="submit" name="submit" value="Submit">
21 </form>
22 David Smith<br>d123<br>AlkalineZ Limited Inc<br>&#039;Chemicals&#039;
23 and &quot;Pharmaceuticals&quot; are our lifeline.<br>male</body>
24 </html>
```

**Figure: Page Source Showing Encoded Data**

# Validating Form Data with PHP [5-5]

## Code Snippet:

```
<?php
// define variables and set to empty values
$nameErr = $pwdErr = $genderErr = $companyErr = "";
$name = $password = $gender = $comment = $company = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = sanitize_data($_POST["name"]);
    }

    if (empty($_POST["password"])) {
        $pwdErr = "Password is required";
    } else {
        $password = sanitize_data($_POST["password"]);
    }
}
```

```
    if (empty($_POST["company"])) {
        $company = "";
    } else {
        $company = sanitize_data($_POST["company"]);
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = sanitize_data($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = sanitize_data($_POST["gender"]);
    }
}
?>
```

Code Snippet shows how error variables store values.



# Display Error Messages [1-3]

## Code Snippet:

```
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
error_reporting(0);
// Initialize variables during creation to empty values
$nameErr = $pwdErr = $genderErr = $companyErr = "";
$name = $pwd = $gender = $comment = $company = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = sanitize_data($_POST["name"]);
    }
    if (empty($_POST["password"])) {
        $pwdErr = "password is required";
```

Code Snippet shows a program to display an error message when the user submits the form without filling the required fields.

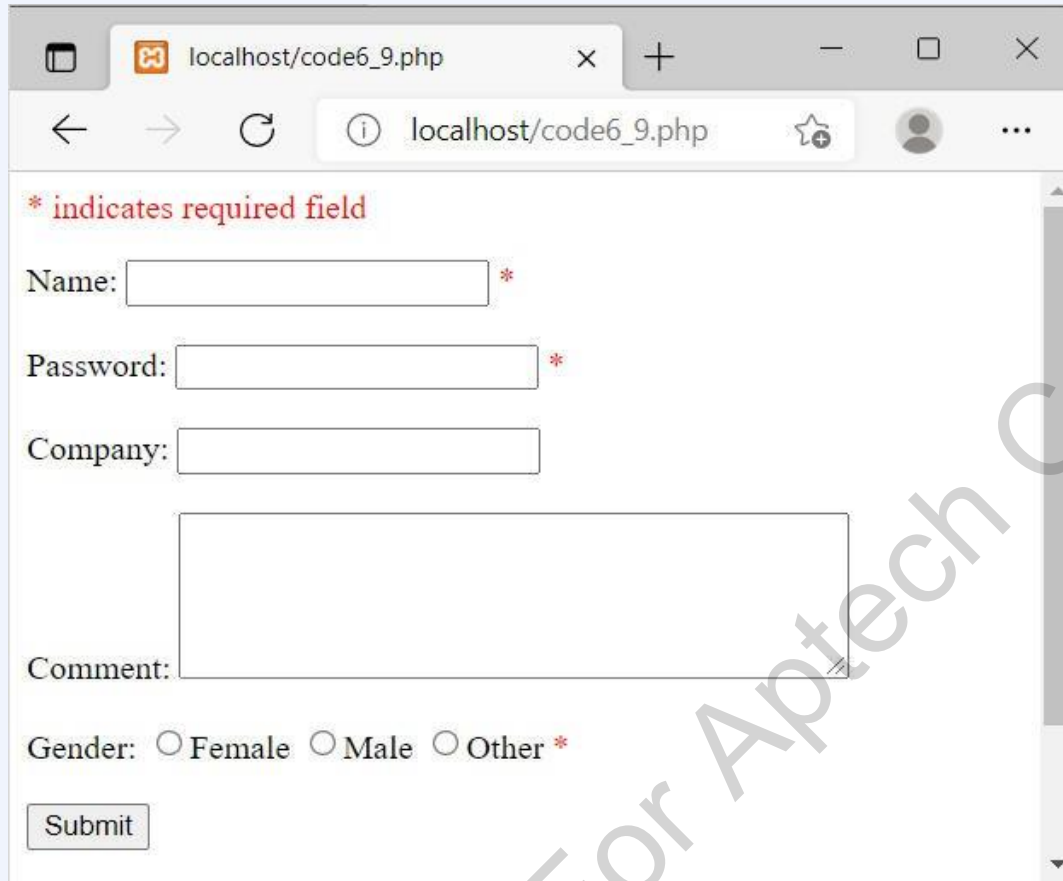
```
    } else {
        $password = sanitize_data($_POST["password"]);
    }
    if (empty($_POST["company"])) {
        $company = "";
    } else {
        $company = sanitize_data($_POST["company"]);
    }
    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = sanitize_data($_POST["comment"]);
    }
    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = sanitize_data($_POST["gender"]);
    }
}
function sanitize_data($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

# Display Error Messages [2-3]

```
?>
<p><span class="error">* indicates required field</span></p>
<form method="post" action="<?php
echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  Password: <input type="password" name="password">
  <span class="error">* <?php echo $pwdErr;?></span>
  <br><br>
  Company: <input type="text" name="company">
  <span class="error"><?php echo $companyErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
```

```
</form>
<?php
echo $name;
echo "<br>";
echo $password;
echo "<br>";
echo $company;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```

# Display Error Messages [3-3]



A screenshot of a web browser window displaying a registration form at localhost/code6\_9.php. The form includes fields for Name, Password, Company, and a Comment box, along with radio buttons for Gender (Female, Male, Other). A 'Submit' button is at the bottom. Red error messages are displayed: '\* indicates required field' at the top, and asterisks next to the Name, Password, and Gender labels. The Gender label also has the text '\* Gender is required' next to it.

\* indicates required field

Name:  \*

Password:  \*

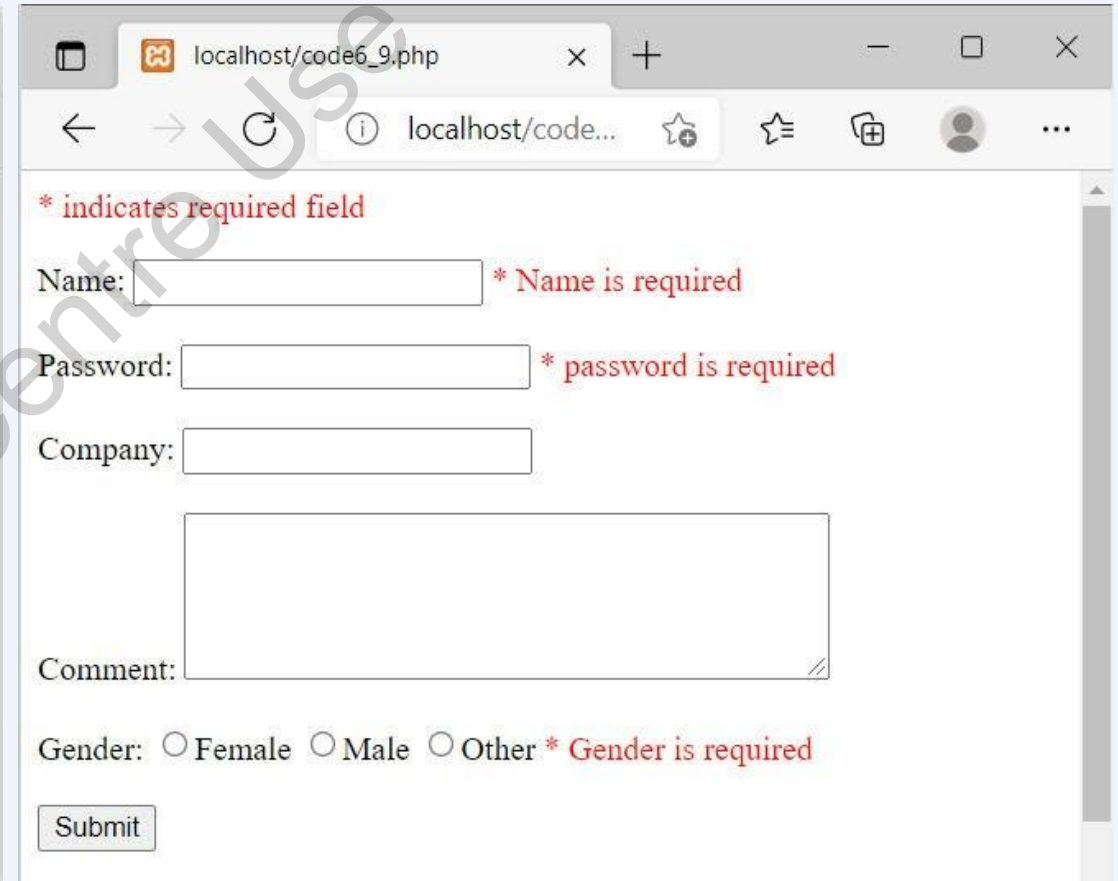
Company:

Comment:

Gender: ☐ Female ☐ Male ☐ Other \*

Submit

**Figure: Page Source Showing Encoded Data**



A screenshot of a web browser window displaying the same registration form as the previous figure. In addition to the general error message and asterisks, specific error messages are shown in red: '\* Name is required' next to the Name field, '\* password is required' next to the Password field, and '\* Gender is required' next to the Gender label.

\* indicates required field

Name:  \* Name is required

Password:  \* password is required

Company:

Comment:

Gender: ☐ Female ☐ Male ☐ Other \* Gender is required

Submit

**Figure: Page Source Showing Encoded Data**

# Validating Against Special Characters [1-2]

In a text field such as name, user can only enter letters and whitespaces. Numbers and special characters are not allowed in the name field.

Following statements check the name field for letters, dashes, apostrophes, and whitespaces:

```
$name = sanitize_data($_POST["name"]);  
if (!preg_match("/^[a-zA-Z-' ]*$/", $name)) {  
    $nameErr = "Only letters and white space  
allowed";  
}
```

An error message is displayed if the value entered for the name field is not valid.

The `preg_match()`

function is used here to search a string for a pattern. It returns **true** if the pattern exists and **false** if the pattern does not exist.

For Antech

# Validating Against Special Characters [2-2]

## Code Snippet:

```
$password = $_POST['password'];
if( strlen($password) < 8 ) {
    $pwdErr .= " Password too short. ";
}
if( strlen($password) > 20 ) {
    $pwdErr .= " Password too long!";
}
if( !preg_match("@[0-9]@", $password) ) {
    $pwdErr .= " Password must include at least one number.";
}
if( !preg_match("#[a-z]+#", $password) ) {
    $pwdErr .= " Password must include at least one letter.";
}
if( !preg_match("#[A-Z]+#", $password) ) {
    $pwdErr .= " Password must include at least one uppercase letter.";
}
if( !preg_match("#[^\w]#", $password) ) {
    $pwdErr .= " Password must include at least one symbol.";
}
if (empty($pwdErr)) {
    if (preg_match("#.*^(?=.{8,20})(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[^\w]).*$#", $password)) {
        echo "Your password is strong.";
    } else {
        echo "Your password is not safe.";
    }
}
```

Code Snippet shows how to check if a password entered by the user is valid or not. An error message is displayed if the password entered is not valid.

# Summary

- Forms can be created using HTML tags and input elements to accept user data and send to a Web server for processing.
- GET and POST methods in PHP can be used in forms to send form data.
- `$_GET` and `$_POST` are superglobal variables and are always accessible, regardless of function, class, or file scope.
- The PHP `$_REQUEST` variable contains values of `$_GET`, `$_POST`, and `$_COOKIE` and is used to get result from form data sent with the GET and POST methods.
- The PHP form input fields have validation rules.
- PHP's `htmlspecialchars()` function can help to encode special characters into HTML entities and can be used to sanitize form data.
- An error message can be generated at the form level or field level through code if user tries to submit a form without filling mandatory fields.
- The `preg_match()` function searches a string for a pattern, returning true if the pattern exists and false otherwise. It is used to validate the name entered.