Faculty of Science and Technology

Department of Computer Science

WEB TECHNOLOGIES (CSC 3222)

Lecture Note 3

Week 3

Supta Richard Philip
Richard@aiub.edu

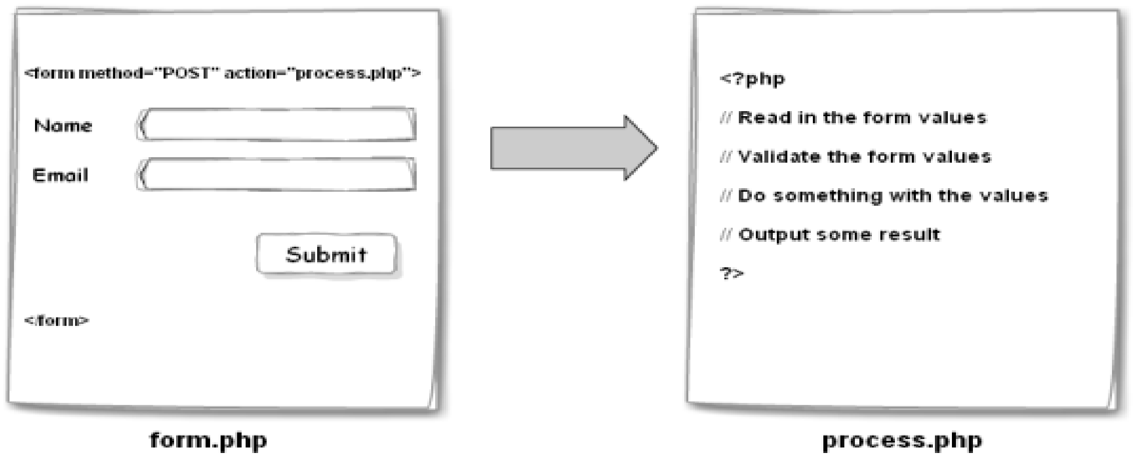
Contents

HTML Form handling

In this note, We will discuss more details about HTML form elements i.e. different type of form, designing different type of HTML form and discuss HTTP GET and POST request as well. Normally Handling form data we use $_GET or $_POST methods and validate those data using PHP. After validation, we will assign them in the class object or array.

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

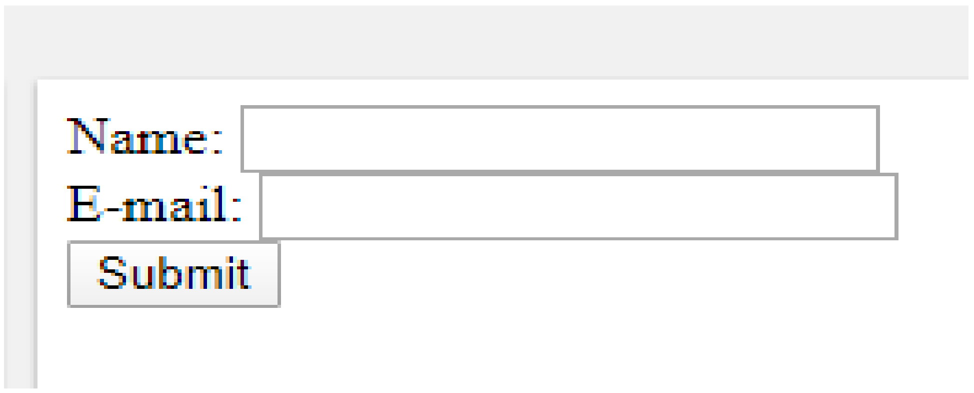
The sequence is as follows:


```
<form method="POST" action="process.php">
```

```
</form>
```
```
<?php
// Read in the form values
// Validate the form values
// Do something with the values
// Output some result
?>
```

Handling form data using HTTP POST

Let's we have the following html form to submit.



Form.php

```
<html>
<body>
<form action="Welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Welcome.php

```
<html>
<body>

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

</body>
</html>
```

Handling form data using HTTP GET

If you have the same form using GET request.

Form.php

```
<html>
<body>
<form action="Welcome.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Welcome.php

```
<html>
<body>

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

</body>
</html>
```

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.

$_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.

Recommendation:

- 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.
- GET may be used for sending non-sensitive data.
- GET should NEVER be used for sending passwords or other sensitive information!
- 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.

Data in Current Page:

- Think SECURITY when processing PHP forms! Proper validation of form data is important to protect your form from hackers and spammers!
- **$_SERVER["PHP_SELF"]** is a super global variable that returns the filename of the currently executing script and sends the submitted form data to the page itself, instead of jumping to a different page.
- This way, the user will also get error messages on the same page as the form.

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

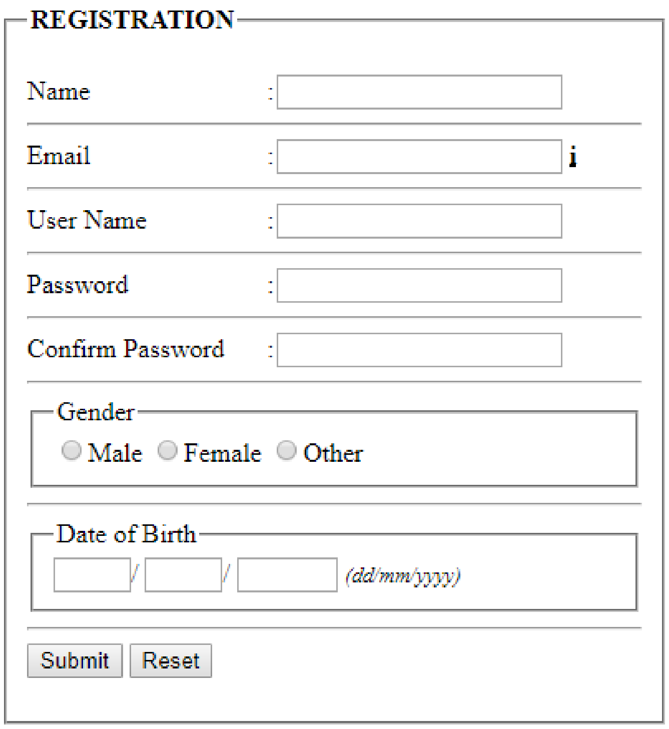
Exercise

1. Design the following login form and registration form and perform the following operations. **Assign the data into the class objects.**

A. Take input and displays that on handler page.

B. Take input and displays that on current page.

C. Take input and displays that on current page such that the input element retains the previous value.

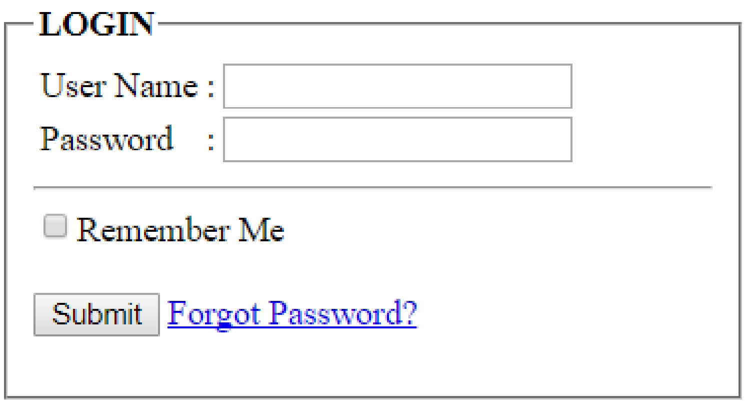


REGISTRATION

Name	:
Email	:
User Name	:
Password	:
Confirm Password	:

Gender
Male Female Other

Date of Birth
/ / (dd/mm/yyyy)



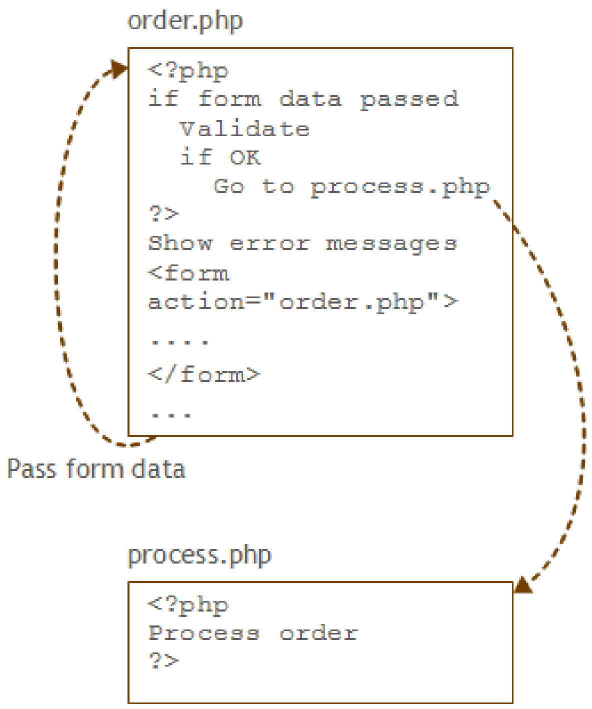
LOGIN

User Name :
Password :

Remember Me

HTML Form Validation

Form data validation is important to protect your form from hackers and spammers. If you want to validate form data, its better to submit form in current page. So use **<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>** in form action and follow the following pattern. After successfully validate jump to other page.



```
<?php
if form data passed
  Validate
  if OK
    Go to process.php
?>
Show error messages
<form
action="order.php">
....
</form>
...
```

```
<?php
Process order
?>
```

To validate form, follow validation rules as follows:

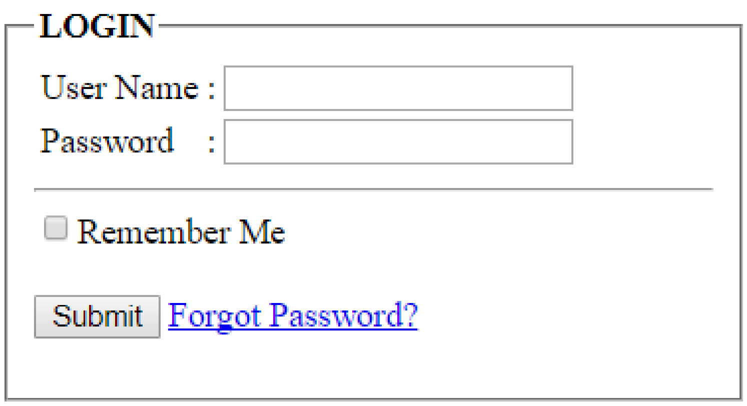
Field	**Validation Rules**
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

Form.php

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

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

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<html>
<body>

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_S
ELF"]);?>">

Name: <input type="text" name="name" value="<?php echo $name;?>">

Email: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?
>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $commen
t;?></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other
<input type="submit" name="submit" value="Submit">

</form>
</body>
</html>
```

Exercise

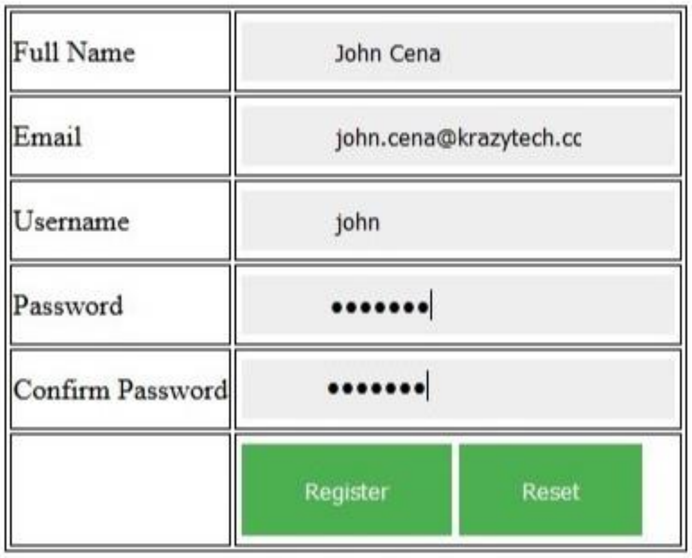
1. Design the following HTML form and perform following validations



Validation Rules

A. User Name can contain alpha numeric characters, period, dash or underscore only
B. User Name must contain at least two (2) characters
C. Password must not be less than eight (8) characters
D. Password must contain at least one of the special characters (@, #, $, %)

2. Design the following HTML form and perform following validations. **After pass the validation, assign the data into the class object.**

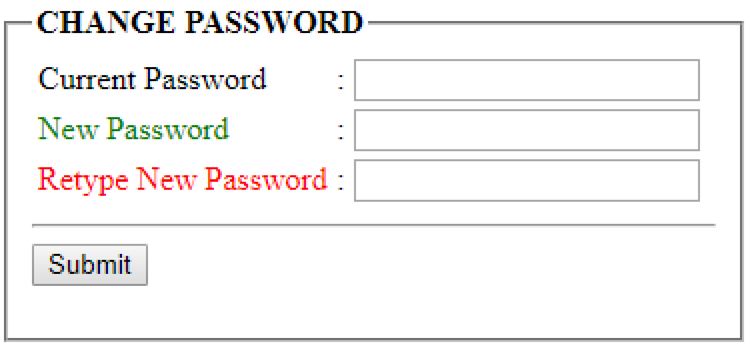


Full Name	John Cena
Email	john.cena@krazytech.cc
Username	john
Password	••••••••
Confirm Password	•••••••
	Register Reset

Validation Rules

- User Name can contain alpha numeric characters, period, dash or underscore only
- User Name must contain at least two (5) characters
- Password must not be less than eight (6) characters
- Password must contain at least one of the special characters (@, #, $, %)
- Password should be same as the confirm Password.
- Must contain a valid email address (with @ and .).
- All fields are required.

3. Design the following HTML form and perform following validations



Validation Rules

A. New Password should not be same as the Current Password

B. New Password must match with the Retyped Password

References

[1] https://www.askaboutphp.com/2010/06/09/php-and-jquery-submit-a-form-without-refreshing-the-page/

[2] https://www.guru99.com/php-forms-handling.html