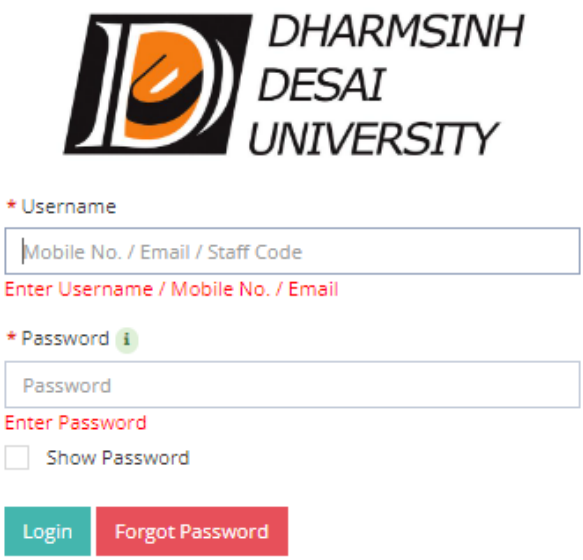
Lab 8

AIM: Validations in PHP

Activation: Have you seen web pages showing descriptive error messages?



Motivation: Why do we need to check for input provided by the user? Why do we need to display descriptive error messages?

When processing a form, it's critical to validate user inputs to ensure data is in a valid format.

There are two types of validations: 1 client-side & 2 server-side:

The client-side validation is performed in the web browsers of the users. To validate data on the client side, you can use HTML5 validation or JavaScript. The client-side validation aims to assist legitimate users in entering data in a valid format before submitting it to the server.

However, client-side validation doesn't prevent malicious users from submitting data that can potentially exploit the application.

The server-side validation validates data in the web server using PHP. To validate data in PHP, you can use the filter_var() and filter_input() functions.

Demonstration:

You'll build an email subscription form that includes a validation feature. The form has the name and email input elements and a submit button.

If you don't enter the name and/or email and click the subscribe button, the form will show the error messages. Also, if you enter an invalid email address, the form will show a different error message.

The following table describes the purpose of each file:

File	Description
index.php	Contains the main logic of the form
header.php	Contains the header code
footer.php	Contains the footer code
get.php	Contains the email subscription form
post.php	Contains the code for handling form submission

Step 1: Check the code of header.php

It contains code for meta and other head elements. It may contain stylesheets and other supporting files.

Step 2: Check the code of footer.php

It contains the enclosing tags that correspond to the opening tags in the header.php file.

Step 3: Check the code of index.php

The index.php file contains the main logic of the form.

How does the index.php work?

First, load code from header.php and footer.php files using the require function at the top and bottom of the file to generate the header and footer.

Second, define the $errors array to store error messages and the $inputs array to store the entered form values. If an input element has invalid data, the index.php will show the entered value stored in the $inputs.

Third, show the form if the HTTP request method is GET by loading the get.php file.

Finally, load the code in the post.php to handle the form submission if the HTTP request method is POST. If the form has any errors, the $errors will not be empty. In this case, show the form again with the error messages stored in the $errors array and entered values stored in the $inputs arrays.

Step 4: Check the code of get.php

The get.php file contains the form.

How does the get.php work?

First, fill the name and email input elements with the entered values stored in the $inputs array only if these values exist.

Second, show the error messages stored in the $errors array if they exist.

Step 5: Check the code of post.php

The post.php validates the form data using the filter_input() and filter_var() functions.

How does it work?

First, define some constants to store the error messages. In a real-world application, you may store all the messages in a separate file:

```
const NAME_REQUIRED = 'Please enter your name';
const EMAIL_REQUIRED = 'Please enter your email';
const EMAIL_INVALID = 'Please enter a valid email';
```

Second, sanitize and validate the name using the filter_input() function. If the name is empty, add an error message to the $errors array.

(Helps: https://www.php.net/manual/en/function.filter-input.php, https://www.php.net/manual/en/filter.filters.php, https://www.php.net/manual/en/filter.filters.sanitize.php)

```
// sanitize and validate name
$name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
$inputs['name'] = $name;

if ($name) {
    $name = trim($name);
    if ($name === '') {
```

```
        $errors['name'] = NAME_REQUIRED;
    }
} else {
    $errors['name'] = NAME_REQUIRED;
}
```

Third, sanitize and validate email using the filter_input() and filter_var() functions. If the email is empty or invalid, add the corresponding error message to the $errors array.

(Helps: https://www.php.net/manual/en/function.filter-input.php, https://www.php.net/manual/en/filter.filters.php, https://www.php.net/manual/en/filter.filters.sanitize.php, https://www.php.net/manual/en/filter.filters.validate.php,)

```
// sanitize & validate email
$email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
$inputs['email'] = $email;
if ($email) {
    // validate email
    $email = filter_var($email, FILTER_VALIDATE_EMAIL);
    if ($email === false) {
        $errors['email'] = EMAIL_INVALID;
    }
} else {
    $errors['email'] = EMAIL_REQUIRED;
}
```

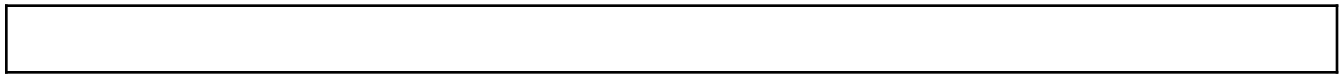
Finally, if the form has no error, show the confirmation message.

Step 6: Execute http://127.0.0.1/lab8/index.php and confirm your understanding.

Engagement:

Assignment

1. Develop the code for achieving results as shown in the following figures.



Details

Give us your honest details to server you better!

Name: Full Name Please enter your name
Email: Email Address Please enter your email
Age: Your correct age Please enter your age
URL: Your website(optional)
IP Address: Your default machine(optior
Registration

Details

Give us your honest details to server you better!

Name: Vini
Email: vini Please enter a valid email
Age: 4 Please enter a valid age value from 18 to 55
URL: Your website(optional)
IP Address: Your default machine(optior
Registration

Details

Give us your honest details to server you better!

Name: Vini
Email: vini@edwar.edu.in
Age: 18
URL: 1234 Please enter a valid URL of your blog
IP Address: 1234 Please enter a valid IP address of your machine
Registration

Details

Give us your honest details to server you better!

Name: Vini
Email: vini@edwar.edu.in
Age: 34
URL: http://ddu.ac.in
IP Address: 192.168.29.152
Registration

Thanks Vini for your details!

Please follow the steps below to grab your vouchers:

1. Check your email (vini@edwar.edu.in) - Find the message sent from support@ourorganization.net
2. Click to confirm - Click on the link in the email to confirm your vouchers.

Integration:

Assignment+

1. What is the importance of validations in a web application?
2. What are the different types of validations?

3. Explain the difference between client-side and server-side validations.
4. What is the use of the filter_var() and filter_input() functions in PHP?
5. Which content normally resides in a header.php file?
6. Which content normally resides in a footer.php file?
7. Which content normally resides in an index.php file?
8. What is the null coalescing operator in PHP? How is it useful?
9. What is the difference between validate and sanitize in PHP?
10. How can we sanitize and validate an email?
11. What is the alternative syntax for control structures in PHP? How is it useful?

Conclusion:
Write in your own words your learning from 👩‍💻(i.e. conceptual understanding/ errors encountered and solutions, etc…)