Web Development

Lecture 6 -Server Side Validation

Outline

- Server side Vs Client Side Validation
- Server side Input Validation in PHP
- Checking for blanks
- Checking for email address
- Checking the length of a field
- Regular Expressions

Validation

Validation comes in three forms,

JavaScript - This is done on the - client side
HTML5 - put directly in the form - client side
PHP - This validation is done on the server side

Validation

JavaScript validation is very easy to hack, as it resides on the client side. People can easily manipulate the JavaScript. Why ? Because code can be copied and changed e.g. remove all javascript validation but keeping the form. Then inputting malicious data to the form.

Client side validation is more about creating a better user experience by informing the user as to errors and how to correct them before moving to the server for processing

Validation

PHP Server side validation is more secure, because it is done on the server side, where there is less opportunity to manipulate the code.

Validation

Validation is one of the most important parts of creating forms on websites.

The user will always make mistakes when inputting into a form!

Validation

When we are creating a form validation, we need to think of the data we want to validate and what combination of

- letters, digits, special characters are allowed
- what is the minimum/maximum length of the form fields.
- Check the data your database is expecting

Validation

We may think of a telephone number as just been made up of digits.

But should we allow different combinations

(087)981-8367

087-9818356

Can we minimise user error e.g. by the use of a drop down menu for date selection (see ryanair web site), country/city selection.

Validation

What happens if the user does not type anything at all? This can happen!

A user may attempt to submit a form without any values.

Feedback

If your form is checking for something, you need to ensure that you print out to the screen to tell the user what went wrong.

If you do not tell the user what went wrong, they will never know and get frustrated with the form.

Validation Functions

What functions can we use to validate our data

String function

isset();
empty();
strlen();

Number functions

is_int()
is_decimal()

More string functions number functions

Examples

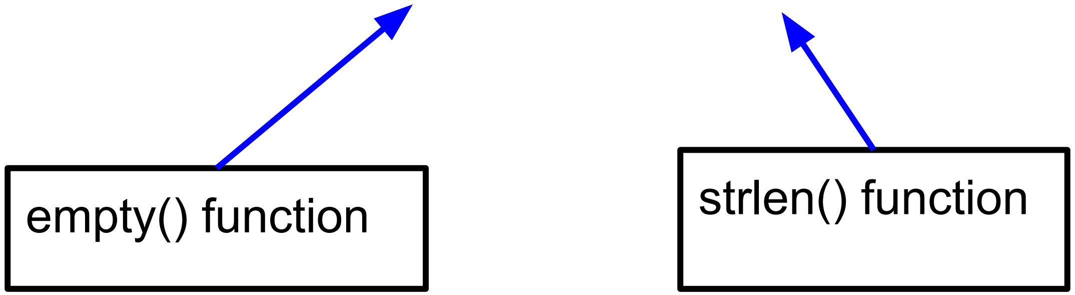
We will now have a look at some examples of validation in PHP using some of PHPs builtin functions.

Open repl.it in a browser

Register.php

We are going to start with validating the **username** on our register form

We want username to be **non blank** and at least **3** characters long



register.php

Additional code in red

```
$usernameErr = "";
$username="";
if($_POST){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $email = $_POST['email'];

        if (empty($username) || strlen($username) < 4 )
              $usernameErr = "Username is required, at least 3 chars";
        if (empty($usernameErr)) {

          try {
```

Error message

We want to add our error message to our form.
We’ll put it just below the input field

We want to keep track of the user name on the form so we will display the value even if the user gets an error, so they have the opportunity to correct the input,

So we add a bit of PHP code to our input field

```
Username <input type="text" name="username" value="<?php echo $username;
?>”/>
```

Ready to print!

Once we have built up our error message, we can then print it out.

```
Username <input type="text" name="username" value="<?php echo $username; ?>"/>
    <span class = "error"> <?php echo $usernameErr;?></span>
```

New Line in register.php

Error String

Register.php

Lets download the In Class files from Moodle, unzip under htdocs and you should have the files in a Lecture6 folder

Run register.php in a browser and test it out

We’ll go thru the code with last weeks homework included

Advanced Checks

Sometimes we may want to check to see if the user has typed in a valid email address.

A valid email address would contain:

1. @ sign,
2. . A dot somewhere in the address
3. Domain address e.g. .com, .org etc.

Advanced Checks

We could make a collection of if statements, checking to see if each of these elements exists in the string….

or we can use an existing function!

filter_var function

Using `filter_var` is incredibly easy. It's simply a PHP function that takes two pieces of data:

- The variable you want to check
- The type of check to use

We can use **filter_var** to validate email data

Advanced Checks - Email

E-mail address variable

Filter we want to apply

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = 'Sorry you did not enter a correct email! ';
}
```

Let's add it to our form with additional check for **$emailErr** in the **if** statement

Ready to print!

Once we have built up our error message, we can then print it out.

```
email<input type="text" name="username" value="<?php echo $email; ?>/>
    <span class = "error">* <?php echo $emailErr;?></span>
```

New Line in register.php

Error String

Advanced Checks - URL

web address
variable

Filter we want to apply

```
if (filter_var($website, FILTER_VALIDATE_URL) )
     { echo 'is valid'; }

else

     { echo 'not valid'; }
```

Length of a field

Let’s make the password field a minimum of 8 chars long using the strlen() function

```
if (strlen($password) < 8) {
   $passwordErr = ‘password is too
   short!’;
}
```

Regular expressions

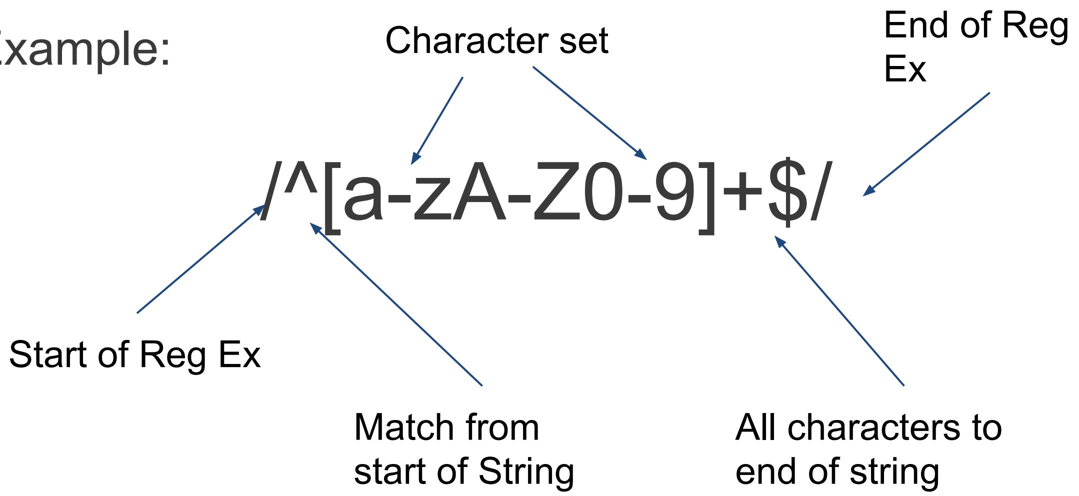
Regular expressions are a language used for parsing and manipulating text. They are often used to perform complex search-and-replace operations, and to ensure that text data is well-formed.

Regular Expressions are like any other language, they require time and effort to learn.

Regular expressions (RegEX) provide a powerful, concise, and flexible means for matching strings of text such as particular characters, words, or patterns of characters.

Regular expressions

Example:

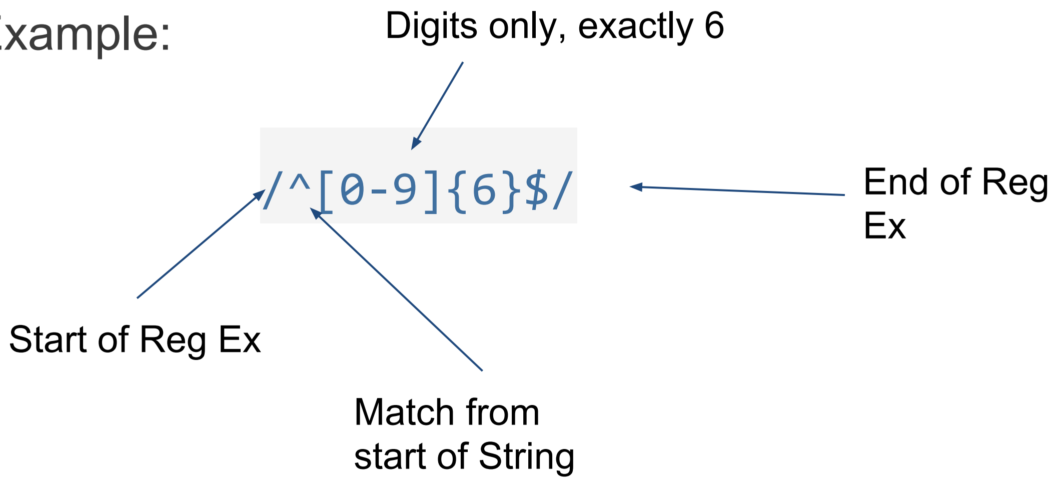


```
/^[a-zA-Z0-9]+$/
```

This will allow any upper or lower case letters as well as numbers.

Regular expressions

Example:



```
/^[0-9]{6}$/
```

This will look for a 6 digit number.

Regular expressions

To use a regular expression we use the php function

preg_match()

If we want to compare a string to a pattern we could call the function as follows

```
// 10. should be invalid
$message="aBc_123";
//
if (preg_match('/^[a-zA-Z0-9]+$/',$message))
   echo "\n10. regx message validates";
else
   echo "\n10. regx message does not validate";
```

Password Validation

Let’s add the reg ex validation to our password field a minimum of 8 chars long letters and digits

```
if (strlen($password) < 8 ||
    !preg_match('/^[a-zA-Z0-9]+$/',$password)
) {
   $passwordErr = ‘password 8 or more
   letters and digits ‘;
}
```

Homework

Add the validation to the **login.php** file. It will be the same as register.php but without email field

Password Validation

Finally we need to update our **if** statement to include the $passwordErr variable

```
 if (empty($usernameErr) && empty($emailErr) && empty($passwordErr)) {


        try { /// continue processing and store to Database
            }catch(PDOException $e) {echo 'Error' . $e;}
 }
```

Next time

We’ll attempt to create a CRUD application with features such as

- password hashing
- logout facility
- Wireframes for page layout