Software Development Tools & Practices

PUSL2020

Level 2

“Validation”

Dr. Rasika Ranaweera (ranaweera.r@nsbm.ac.lk | +94 11 544 6126)

Faculty of Computing| NSBM Green University

What is form validation?

validation: ensuring that form's values are correct

some types of validation:

- preventing blank values (email address)
- ensuring the type of values
- integer, real number, currency, phone number, Social Security number, postal address, email address, date, credit card number, ...
- ensuring the format and range of values (ZIP code must be a 5-digit integer)
- ensuring that values fit together (user types email twice, and the two must match)

Become a Registered User!

Get access to WeightWatchers.com's buzzing Community:

- Start a blog
- Participate in a Challenge
- Save your favorite recipes
- Post to our boards

WeightWatchers

Important message: Please review and correct the information below.

Create Your Registered User Account Login

First name: Xenia

Last name:
Please enter your last name. It's required information.

Your birthdate: Month Day Year
Please select your month, day, and year of birth. It's required information.

Your gender: Female Male
Please enter your gender. It's required information.

State: (Select One)
Please choose a state. It's required information.

Zip code
Please enter a 5-digit ZIP code. It's required information.

E-mail:
Please enter your email address in the following format: abc@example.com. It's required information.

Re-enter e-mail:

Client vs. server-side validation

Validation can be performed:

- **client-side** (before the form is submitted)
  - *can lead to a better user experience, but not secure (why not?)*
- **server-side** (in PHP code, after the form is submitted)
  - *needed for truly secure validation, but slower*
- both
  - best mix of convenience and security, but requires most effort to program

An example form to be validated

```
<form action="http://foo.com/foo.php" method="get">
	<div>
		City: <input name="city" /> <br />
		State: <input name="state" size="2" maxlength="2" /> <br
/>
		ZIP: <input name="zip" size="5" maxlength="5" /> <br />
		<input type="submit" />
	</div>
</form>
```

HTML

Let's validate this form's data on the server...

Basic server-side validation code - PHP

```
<?php
$city = $_REQUEST["city"];
$state = $_REQUEST["state"];
$zip = $_REQUEST["zip"];
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
?>
      <h2>Error, invalid city/state submitted.</h2>
<?php
}
?>
```

PHP

basic idea: examine parameter values, and if they are bad, show an error message and abort

Basic server-side validation code

validation code can take a lot of time / lines to write

- How do you test for integers vs. real numbers vs. strings?
- How do you test for a valid credit card number?
- How do you test that a person's name has a middle initial?
- How do you test whether a given string matches a particular complex format?

Basic Regular Expression

```
/abc/
```

in PHP, regexes are strings that begin and end with /

the simplest regexes simply match a particular substring

the above regular expression matches any string containing "abc":

YES: "abc", "abcdef", "defabc", ".=.abc.=.", ...

NO: "fedcba", "ab c", "PHP", ...

Wildcards

A dot . matches any character except a \n line break

"/.oo.y/" matches "Doocy", "goofy", "LooNy", ...

A trailing i at the end of a regex (after the closing /) signifies a case-insensitive match

"/xen/i" matches “Xenia", “xenophobic", “Xena the warrior princess", “XEN technologies” ...

Special characters: |, (), ^, \

| means *OR*

"/abc|def|g/" matches "abc", "def", or "g"

There's no *AND* symbol. Why not?

() are for grouping

"/(Homer|Marge) Simpson/" matches "Homer Simpson" or "Marge Simpson"

^ matches the beginning of a line; $ the end

"/^<!--$/" matches a line that consists entirely of "<!--"

Special characters: |, (), ^, \

\ starts an escape sequence

many characters must be escaped to match them literally: / \ $. [ ] ( ) ^ * + ?

"/<br \/>/" matches lines containing
 tags

Quantifiers: *, +, ?

* means 0 or more occurrences
  "/abc*/" matches "ab", "abc", "abcc", "abccc", ...
  "/a(bc)*/" matches "a", "abc", "abcbc", "abcbcbc", ...
  "/a.*a/" matches "aa", "aba", "a8qa", "a!?_a", ...

+ means 1 or more occurrences
  "/a(bc)+/" matches "abc", "abcbc", "abcbcbc", ...
  "/Goo+gle/" matches "Google", "Gooogle", "Goooogle", ...

? means 0 or 1 occurrences
  "/a(bc)?/" matches "a" or "abc"

More quantifiers: {min,max}

{min,max} means between min and max occurrences (inclusive)

"/a(bc){2,4}/" matches "abcbc", "abcbcbc", or "abcbcbcbc"

min or max may be omitted to specify any number

{2,} means 2 or more

{,6} means up to 6

{3} means exactly 3

Character sets: []

[] group characters into a character set; will match any single character from the set

"/[bcd]art/" matches strings containing "bart", "cart", and "dart"

equivalent to "/(b|c|d)art/" but shorter

inside [], many of the modifier keys act as normal characters

"/what[!*?]*/" matches "what", "what!", "what?**!", "what??!",

Character ranges: [start-end]

inside a character set, specify a range of characters with -

"/[a-z]/" matches any lowercase letter

"/[a-zA-Z0-9]/" matches any lower- or uppercase letter or digit

an initial ^ inside a character set negates it

"/[^abcd]/" matches any character other than a, b, c, or d

Character ranges: [start-end]

inside a character set, - must be escaped to be matched

"/[+\-]?[0-9]+/" matches an optional + or -, followed by at least one digit

Escape sequences

special escape sequence character sets:

- \d matches any digit (same as [0-9]); \D any non-digit ([^0-9])
- \w matches any “word character” (same as [a-zA-Z_0-9]); \W any non-word

char

- \s matches any whitespace character (, \t, \n, etc.); \S any non-whitespace

Regular expressions in PHP

regex syntax: strings that begin and end with /, such as "/[AEIOU]+/"

preg_match(regex, string)	returns TRUE if string matches regex
preg_replace(regex, replacement, string)	returns a new string with all substrings that match regex replaced by replacement
preg_split(regex, string)	returns an array of strings from given string broken apart using the given regex as the delimiter (similar to explode but more powerful)

Regular expressions example

```
echo preg_match ('/test/', "a test of preg_match");
echo preg_match ('/tutorial/', "a test of preg_match");

$matchesarray[0] = "http://www.tipsntutorials.com/"
$matchesarray[1] = "http://"
$matchesarray[2] = "www.tipsntutorials.com/"
preg_match ('/(http://)(.*)/', $matchesarray)
```

PHP

Regular expressions example

```
# replace vowels with stars
$str = "stay hungry stay foolish";
$str = preg_replace("/[aeiou]/", "*", $str);
#

# break apart into words
$words = preg_split("/[ ]+/", $str);
#

# capitalize words that had 2+ consecutive vowels
for ($i = 0; $i < count($words); $i++) {
if (preg_match("/\\*{2,}/", $words[$i])) {
$words[$i] = strtoupper($words[$i]);
}
} #
```

PHP

Regular expressions example

```php
# replace vowels with stars
$str = "the quick brown fox";
$str = preg_replace("/[aeiou]/", "*", $str);
# "th* q**ck br*wn f*x"

# break apart into words
$words = preg_split("/[ ]+/", $str);
# ("th*", "q**ck", "br*wn", "f*x")

# capitalize words that had 2+ consecutive vowels
for ($i = 0; $i < count($words); $i++) {
if (preg_match("/\\*{2,}/", $words[$i])) {
$words[$i] = strtoupper($words[$i]);
}
} # ("th*", "Q**CK", "br*wn", "f*x")
```

PHP

Why validate?

- XSS (Cross Site Scripting)

Code	Sample
Application code	<input value="userInput”>
Malicious string	"><script>...</script><input value=”
Resulting code	<input value=""><script>...</script><input value="">

- SQLI (SQL Injection)

Code	Sample
Application code	SELECT * FROM Users WHERE Username='$username' AND Password='$password'
Malicious string	$username = 1' or '1' = '1
Resulting code	SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'