



Lecture 10

Form Validations and Regular Expression

SE-805 Web 2.0 Programming (supported by Google)

<http://my.ss.sysu.edu.cn/courses/web2.0/>

School of Software, Sun Yat-sen University

Outline

- **Form Validations**
- Regular Expression

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)

A real Form that Uses Validation


[← Cancel](#)

Secure Site

Questions? Call us:

(800) 788-7000

Signing Up is Easy

- Some of the information you entered is missing or incorrect. Please check all highlighted messages below.
- Please enter Last Name using letters, apostrophes or dashes.
 - Enter a valid date for Date of Birth.
 - Please enter a valid e-mail address.

Personal Info

 First Name:

 Last Name:

 Date of Birth:

 E-mail Address:

- Identify yourself by your:
- Account Number
 - ATM/Debit Card
 - Credit Card

Step 1

CONTINUE →

Client vs. Server 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 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

City:

State:

ZIP:

output

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

Basic Server-Side Validation Code

```
$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
- 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?)

Outline

- Form Validations
- **Regular Expression**

What is a Regular Expression?

```
"/^[a-zA-Z_\-]+@(([a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}$/"
```

- **Regular expression** ("regex"): a description of a pattern of text
 - Can test whether a string matches the expression's pattern
 - Can use a regex to search/replace characters in a string
- Regular expressions are extremely powerful but tough to read (**the above regular expression matches email addresses**)
- Regular expressions occur in many places:
 - Java: **Scanner**, String's **split** method
 - Supported by PHP, JavaScript, and other languages
 - Many text editors (**Notepad++**, **TextPad**) allow regexes in search/replace

Basic Regular Expressions

```
"/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
 - `/mart/i` matches "Marty Stepp", "smart fellow", "WALMART", ...

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 "<!--"
- \ starts an escape sequence
 - Many characters must be escaped to match them literally: `/\ $. [] () ^ * + ?`
 - `"/<br V>/"` 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", "Goooogle", "Goooooogle", ...
- **?** 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??!", ...
- What regular expression matches DNA (strings of A, C, G, or T)?

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
- Inside a character set, `-` must be escaped to be matched
`"/[+\-]?[0-9]+/"` matches an optional + or -, followed by at least one digit
- What regular expression matches letter grades such as A, B+, or D- ?
- `"/[ABCDF][+\-]?/"`

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
- What regular expression matches dollar amounts of at least \$100.00 ?
 - **"^\d{3,}\.\d{2}/"**

Regular Expressions in PHP (PDF)

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

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

Regular Expression Example

```
# 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
```

- Notice how `\` must be escaped to `\\`

PHP Form Validation with Regexes

```
$state = $_REQUEST["state"];  
if (!preg_match("/[A-Z]{2}/", $state)) {  
?>  
  
    <h2>Error, invalid state submitted.</h2>  
  
<?php  
}
```

PHP

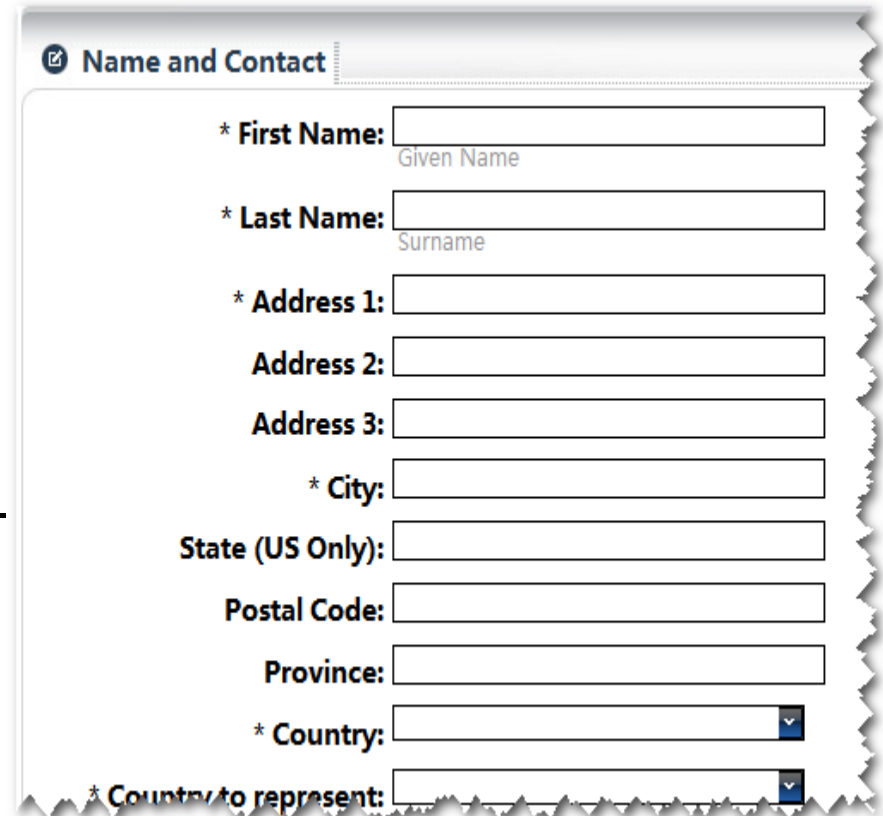
- Using `preg_match` and well-chosen regexes allows you to quickly validate parameters
- Interestingly, we often **DON'T** want to give a very descriptive error message here (why?)

Summary

- Form Validations
 - Client vs. server
 - Basic validations code
- Regular Expression
 - . | () \
 - * + ?
 - {min, max}
 - [] [start-end]
 - \d \D \w \W \s \S
 - Regular expressions in PHP

Exercises

- Write a php page to mimic the registration page of Topcoder
 - Get initial html from <https://www.topcoder.com/reg/>
 - Convert it to a php page, which **only** shows data you submitted at the top of this page when all your data are valid, otherwise shows error messages instead
 - All fields marked with * are none-blank
 - A postal code is a 6 digits number
 - Use proper regular expressions for validations



The image shows a registration form titled "Name and Contact" with the following fields:

- * First Name: (Label: Given Name)
- * Last Name: (Label: Surname)
- * Address 1:
- Address 2:
- Address 3:
- * City:
- State (US Only):
- Postal Code:
- Province:
- * Country: (Dropdown menu)
- * Country to represent: (Dropdown menu)

Further Readings

- PHP Regular Expression tutorials:
- <http://www.phpro.org/tutorials/Introduction-to-PHP-Regex.html>
- http://www.webcheatsheet.com/php/regular_expressions.php
- PHP Regular Expression examples:
http://www.roscripts.com/PHP_regular_expressions_examples-136.html

Thank you!

