

FORM VALIDATION

Regular Expressions

Announcements

2

- First test will be on Wednesday, March 5 during lecture.
- Master branch on betaweb was tested last night and is good to go tomorrow.
- For the final project, you may partner with students from CSC 296 (Database Systems) to do the database design and implementation.
- A tutorial on Software Architecture for Web Programming is available on the course web site.

First Test

3

- All questions will be about Web Programming
 - ▣ Infrastructure: AWS, Unix, Apache, Git, ...
 - ▣ HTML
 - ▣ CSS
 - ▣ PHP
 - ▣ JavaScript
- You may bring on 8.5x11 (or A4) sheet of paper with notes.
- There will be different versions for different seats.

Final Project Database

4

- The professor in CSC 296: Database Systems, Dr. Koomen, is interested in giving his students the opportunity to work with a real projects.
 - ▣ Students from that class will be contacting the Scrum Master to offer their services.
 - ▣ Dealing with databases is complicated, and these students will be able to provide real helpSW

Architecture

5

- Architecture is high level design that specifies the interfaces of the modules that make up a program.
- Three level architecture for the web:
 - ▣ User interface
 - Front end; client/browser
 - Deals with events caused by user
 - HTML, CSS, JavaScript
 - ▣ Business Logic
 - Back end; server
 - Deals with

Three Level Architecture

6

- User interface
 - ▣ Front end; client/browser
 - ▣ Deals with events caused by user
 - ▣ HTML, CSS, JavaScript
- Business Logic
 - ▣ Back end; server
 - ▣ Input, Process, Output
 - ▣ PHP, Java, Ruby

Three Level Architecture

7

- Database (DBMS)
 - ▣ Back end; Server
 - ▣ Manages the data the program uses
 - ▣ SQL; PHP

8

Form Validation

What is form validation?

9

- **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

10

The screenshot shows a Firefox browser window with several tabs open. The active tab is 'WeightWatchers.com: Si...'. The address bar shows the URL 'https://signup.weightwatchers.com/SignupVersions/registration/StepOne.aspx'. The page content includes a sidebar on the left with the heading 'Become a Registered User!' and four links: 'Start a blog', 'Participate in a Challenge', 'Save your favorite recipes', and 'Post to our boards'. The main content area is titled 'Create Your Registered User Account Login' and contains a registration form with the following fields and validation messages:

- First name:** Xenia
- Last name:** [Empty] **X**
Please enter your last name. It's required information.
- Your birthdate:** Month [Dropdown] Day [Dropdown] Year [Dropdown] **X**
Please select your month, day, and year of birth. It's required information.
- Your gender:** Female Male **X**
Please enter your gender. It's required information.
- State:** (Select One) [Dropdown] **X**
Please choose a state. It's required information.
- Zip code:** [Empty] **X**
Please enter a 5-digit ZIP code. It's required information.
- E-mail:** [Empty] **X**
Please enter your email address in the following format: abc@example.com. It's required information.
- Re-enter e-mail:** [Empty]

At the bottom of the browser window, there is a search bar with the text 'Find: prereq' and navigation buttons for 'Next', 'Previous', 'Highlight all', and 'Match case'.

Client vs. server-side validation

11

- 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

12

```
<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

13

```
$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

14

- 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?

15

Standup

Discuss questions with your Scrum Team

Regular expressions

16

```
[a-z]at           #cat, rat, bat...
[aeiou]
[a-zA-Z]
[^a-z]           #not a-z
[[:alnum:]]+     #at least one alphanumeric char
(very) *large    #large, very very very large...
(very){1, 3}     #counting "very" up to 3
^bob             #bob at the beginning
com$            #com at the end
```

PHPRegExp

- Regular expression: a pattern in a piece of text
- PHP has:
 - POSIX
 - **Perl regular expressions**

Delimiters

17

```
/[a-z]/at          #cat, rat, bat...  
#[aeiou]#  
/[a-zA-Z]/  
~[^a-z]~          #not a-z  
/[[:alnum:]]+/  
#(very) *#large   #large, very very very large...  
~(very){1, 3}~    #counting "very" up to 3  
/^bob/            #bob at the beginning  
/com$/            #com at the end  
  
/http:\\/  
// #http://#      #better readability
```

PHPRegExp

- Used for Perl regular expressions (preg)

Basic Regular Expression

18

```
/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

19

- 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: |, (), ^, \

20

- | 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: |, (), ^, \

21

- \ starts an escape sequence
 - ▣ many characters must be escaped to match them
literally: / \ \$. [] () ^ * + ?
 - ▣ "/<br \/>/" matches lines containing
 tags

Quantifiers: *, +, ?

22

- * 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}

23

- {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: []

24

- [] 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]

25

- 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]

26

- 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- ?

Escape sequences

27

- 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 ?

Regular expressions in PHP (PDF)

28

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

function	description
<code><u>preg_match</u>(regex, string)</code>	returns TRUE if string matches regex
<code><u>preg_replace</u>(regex, replacement, string)</code>	returns a new string with all substrings that match regex replaced by replacement
<code><u>preg_split</u>(regex, string)</code>	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

29

```
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://)(.*)/', "http://  
www.tipsntutorials.com/", $matchesarray)
```

PHP

Regular expressions example

30

```
# 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

PHP form validation w/ regexes

31

```
$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 query parameters against complex patterns

Another PHP experiment

32

- Write a PHP script that tests whether an e-mail address is input correctly. Test using valid and invalid addresses
- Use array
- Use function