CSC 2720
Building Web Applications

PHP
PERL-Compatible Regular Expressions
Regular Expressions

- For defining patterns to match strings

- PHP supports two kinds of regular expressions
  - Regular Expressions (Perl-Compatible)
  - Regular Expression (POSIX Extended)
## Meta-characters inside a regular expression

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>For escaping special characters</td>
</tr>
<tr>
<td>^</td>
<td>Matches the beginning of a string</td>
</tr>
<tr>
<td>$</td>
<td>Matches the end of a string</td>
</tr>
<tr>
<td>.</td>
<td>Matches any single character except newline</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[</td>
<td>Start of a class</td>
</tr>
<tr>
<td>]</td>
<td>End of a class</td>
</tr>
<tr>
<td>(</td>
<td>Start of a sub-pattern</td>
</tr>
<tr>
<td>)</td>
<td>End of a sub-pattern</td>
</tr>
<tr>
<td>{</td>
<td>Start of a qualifier</td>
</tr>
<tr>
<td>}</td>
<td>End of a qualifier</td>
</tr>
</tbody>
</table>
Regular Expression Examples

- /ab/  -- matches any string containing "ab"
  - e.g., cab, fabulous

- /^ab/  -- matches any string that begins with "ab"
  - e.g., absolute, abnormal, abc

- /ab$/  -- matches any string that ends with "ab"
  - e.g., cab, matlab

- /^ab$/  -- matches only the string "ab"

- /yes|no/  or  /(yes)|(no)/
  - both match either "yes" or "no"
Regular Expression Examples

- `/^c(a|u|ur)b$/` -- matches "cab", "cub", or "curb"
- `/^c.b$/` -- matches "cab", "cub", "cxb", "c3b", etc.
- `\` -- matches a single back slash character in a string.
  - When written as a PHP string, you need to write "\\"
## Quantifiers – To specify the quantity

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>0 or 1</td>
</tr>
<tr>
<td>*</td>
<td>0 or more</td>
</tr>
<tr>
<td>+</td>
<td>1 or more</td>
</tr>
<tr>
<td>{x}</td>
<td>Exactly $x$ occurrences</td>
</tr>
<tr>
<td>{x, y}</td>
<td>Between $x$ and $y$ occurrences (inclusive)</td>
</tr>
<tr>
<td>{x, }</td>
<td>At least $x$ occurrences</td>
</tr>
</tbody>
</table>
Regular Expression Examples

- `/^a*b$/` -- matches "b", "ab", "aab", "aaaaab", etc.
- `/^a.*b$/` -- matches any string that begins with 'a' and ends with 'b'
- `/c.+t/` -- matches any substring with at least one characters between 'c' and 't'
  - e.g., "cat", "cart", "comet", "ccccccccct", etc. (but not "ct")
- `^(0|1|2|3|4|5|6|7|8|9){5}$/` -- Matches any 5-digit number
Character Classes

- A class is created by placing characters within square brackets ("["] and "]"). The resulting pattern matches any single character belong to the class.
  
  e.g.,
  - [aeiou] – matches any lowercase vowel
  - [ABCabc] – matches 'A', 'B', 'C', 'a', 'b', or 'c'

- You can use hyphen '-' to specify a range of characters within the square brackets.
  
  e.g.
  - [A-Z] – matches any of the uppercase letters
  - [A-Za-z0-9] – matches any of the alphanumerical characters

- '^' is a negation operator when used as the first character in the class.
  
  e.g., [^0-9] – matches any character that is not a digit
## Commonly Used Character Classes

<table>
<thead>
<tr>
<th>CLASS</th>
<th>SHORTCUT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0-9]</td>
<td>\d</td>
<td>Any digit</td>
</tr>
<tr>
<td>[\f\r\t\n\v]</td>
<td>\s</td>
<td>Any white space</td>
</tr>
<tr>
<td>[A-Za-z0-9_]</td>
<td>\w</td>
<td>Any word character (alphanumeric and '_')</td>
</tr>
<tr>
<td>[^0-9]</td>
<td>\D</td>
<td>Not a digit</td>
</tr>
<tr>
<td>[^\f\r\t\n\v]</td>
<td>\S</td>
<td>Not a white space</td>
</tr>
<tr>
<td>[^A-Za-z0-9_]</td>
<td>\W</td>
<td>Not a word character</td>
</tr>
</tbody>
</table>

### Example:

- `/^[0-9]{5}$/` or `/^[\d]{5}$/`
  - Matches any 5-digit number
Modifiers

- These characters, when placed after the closing delimiter, alter the behavior of a regular expression.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Anchors the pattern to the beginning of the string</td>
</tr>
<tr>
<td>i</td>
<td>Enable case-insensitive mode</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline matching</td>
</tr>
<tr>
<td>s</td>
<td>Has the period match every character, including newline</td>
</tr>
<tr>
<td>x</td>
<td>Ignores most white space</td>
</tr>
<tr>
<td>U</td>
<td>Performs a non-greedy match (i.e., matches as little data as possible)</td>
</tr>
</tbody>
</table>

- A more complete list can be found at
Regular Expression Examples

- `/\$for.*/i`
  - Matches any string that starts with "for" regardless of letter case

- `/<.+>/`
  - matches the longest substring that starts with '<' and ends with '>' (because of greedy match)

- `/<.+?>/  or  /<.+>/U  or  /<[\^>]+>/`
  - matches any HTML tag in a string
  - A '?' after a quantifier indicates "non-greedy match" is to be used.
// Check if a user's name is made up of 8-12 alphanumeric characters
if (preg_match('/^[A-Za-z0-9]{8,12}$/', $username)) {
    // OK
}

// The pattern /a[\S]*/ matches a substring that starts with 'a' and follows by any number of non-white space characters.
if (preg_match('/a[\S]*/', "x abc faa axx", $match)) {
    // $match is the first substring that matches the pattern
    echo $match;     // Output abc
}

if (preg_match_all(  
    '/a[\S]*/',  
    "x abc faa axx",  
    $matches) {  
    // $matches becomes an array containing "abc", "aa", and "axx"
}
PHP Functions and Examples

```php
$str = preg_replace('/dog/', 'cat', 'I love dog.'); // $str becomes "I love cat.";

// Using backreferences followed by numeric literals
$string = 'April 15, 2003';
$pattern = '/(\w+) (\d+), (\d+)/i';
$replacement = '${1}1,$3';
$out = preg_replace($pattern, $replacement, $string);
// $out becomes "April1,2003"

// Strips excess whitespace from a string.
$str = 'foo o';
$str = preg_replace('/\s\s+/', ' ', $str);
// $str becomes "foo o"

// Split the phrase by any number of commas or white space characters
$keywords = preg_split("/\[\s,\]+/", "foo , bar, baz");
// keywords becomes an array containing "foo", "bar", "baz"
```
References and Related Resources

- Regular Expressions (Perl-Compatible)