eZ systems

Regular expression

- The magic behind text

Share your information

eZ systems

Share your information

Speaker

- Kore Nordmann
  - Working as a Software Developer for eZ systems
  - PEAR maintainer / developer
- Software projects
  - Image_3D
  - eZ components

eZ systems

Share your information

Content

- What are regular expressions?
- Standards
- Basic search patterns
- Usage in PHP
- Extended search patterns
- Examples

eZ systems

1) What are regular expressions

Share your information

eZ systems

Share your information

1.1) Theory

- A regular expression describes a language
- A language is a amount of words
- A word consist of any characters

eZ systems

1.2) Usage of regular expressions

- A typical question:
  - Is foo@bar.de an email address?
- Regular expression are often used to verify if a word is member of a desired language
- Regular expressions are rarely used to describe a language

Share your information

eZ systems

2) Standards

Share your information

eZ systems

Share your information

2.1) POSIX

- Regular expressions following the POSIX standard 1003.2
- Available in PHP by the ereg* functions
- Deprecated:
  - Slow
  - Outdated
  - Not binary safe

eZ systems

Share your information

2.2) PCRE

- Perl Compatible Regular Expressions
- Available in PHP by the preg_* functions
- Faster and more powerful regular expressions

eZ systems

Share your information

2.3) Theoretical background

- A subgroup of the patterns used in POSIX and PCRE
- Simple conversion to and from Final State Machines
  - Regular languages (Chomsky Type 3)
- Every POSIX compatible regular expression can be converted
- PCRE regular expressions can not be converted

eZ systems

3) Basic patterns

Share your information

eZ systems

Share your information

3.1) The first regular expression (1)

- . Any character
- ? One or none occurrence of subpattern
- + At minimum one occurrence of subpattern
- * Any count of occurrences of subpattern

eZ systems

3.1) The first regular expression (2)

- /pattern/modifier
- For the delimiter each character is allowed except
  - Alphanumeric chars
  - Backslash
- Established characters for the delimiter
  - / # @ ~
- Example for a regular expression
  - /./

Share your information

eZ systems

3.1) The first regular expression (3)

- Which language is defined by this regular expression?
  - What are the strings validated by this regular expression?
- Each string which is one character long – the type of the char does not matter.
- No empty strings
- Each string containing something of any size

Share your information

eZ systems

3.1) The first regular expression (4)

- Exactly one character
  - /^.$/
- One or more characters
  - /^.+$/

Share your information

eZ systems

3.1) The first regular expression (5)

- Matching more then one character
  - {x} x occurrences of subpattern
  - {x,} at minimum x occurrences of subpattern
  - {x,y} at minimum x and at maximum y occurrences of subpattern
- /^.{3,}$/
  - Will match each string which is at minimum 3 chars long

Share your information

eZ systems

Share your information

3.2) Character classes (1)

- FAT file system style filenames
  - /^.{1,8}\.[a-z]{3}$/i
- Character classes are sets of characters which will be accepted
- Examples
  - [a-z] => [abcdefghijklmnopqrstuvwxyz]
  - [0-9] => [0123456789]
  - [a-z0-9_-] => [abcdefghijklmnopqrstuvwxyz0123456789_-]
- The char – is expected to be the first or last character in a character class

eZ systems

Share your information

3.2) Character classes (2)

- All numbers from 13 to 68
- Wrong:
  - [13-68]
  - Would be: [134568]
- Correct
  - /^(1[3-9]|[2-5][0-9]|6[0-8])$/

eZ systems

Share your information

3.3) Inverted character classes

- /^[^0-9]+$/
- Strings containing no numbers
- A ^ as the first character of character class inverts the matching characters of a character class

eZ systems

3.4) A bit more complex example

- /([a-z0-9_]{1,8})\.(txt|php)/
- Brackets allow grouping of subpatterns
- A pipe in bracket works like a conditional or

Share your information

eZ systems

4) Usage in PHP

Share your information

eZ systems

Share your information

4.1) List of PCRE functions

- preg_match()
- preg_match_all()
- preg_split()
- preg_replace()

eZ systems

4.2) Stringhandling (1)

- Handling of backslashes in PHP code
- PHP-Code:
  - <?php echo '/\.\\.\\\.\\\\./'; ?>
- Resulting output
  - /\.\.\\.\\./

Share your information

eZ systems

4.2) Stringhandling (2)

- Wanted regular expression
  - /[\\\{\}]/
- First wrong try
  - <?php preg_match('**/[\\\{\}]/**', $string);
- Processed regular expression
  - /[\\{\}/
- Correct usage
  - <?php
  - preg_match('**/[\\\\\\{\\}]/**', $string);
  - ?>

Share your information

eZ systems

5) Extended search patterns

Share your information

eZ systems

5.1) Greedy patterns (1)

- Examples use the following string:
  - Hello world, how are you?
- Regular expressions are greedy by default
- /.*/ will match the complete string

Share your information

eZ systems

5.2) Greedy patterns (2)

- /(.*)<\/b>/
  - world, how are you
- /(.*)<\/b>/U
  - world
- /(.*?)<\/b>/
  - world
- /(.*?)<\/b>/U
  - world, how are you

Share your information

eZ systems

5.3) Backreferences

- It is possible to access matches of subpatterns in a regular expression
  - Hello world, how are you?
  - /<(\w+)>(.*?)<\/\1>/
  - array(b, i), array(world, you)

Share your information

eZ systems

Share your information

5.4) Recursion

- Recursive structures are not detectable or matchable
  - It is impossible to check if there is a closing bracket for each opening bracket
- Example:
  - <?php
  - $text = 'Hello world, how are you?';
  - echo preg_replace(**'/<([a-z]+)\\s*>(.*?)<\\/\\1\\s*>/is'**, '{\\1}\\2{/\\1}', $text);
  -
  - {b}Hello world{/b}, how {i}are{/i} you?

eZ systems

Share your information

5.5) Line breaks (1)

- Matching multiline text with regular expressions may be a bit confusing
- Text:
  - Hello world
  - FrOSCon
- preg_match_all('**/(.+)/**', $text, $matches);
- Matches:
  - array('Hello world', 'FrOSCon')
- The . matches no line breaks by default
- The modifier **s** changes this behavior
  - /(.+)/s

eZ systems

Share your information

5.5) Line breaks (2)

- /^(.+)$/
  - >> NULL
- /^(.+)$/m
  - array(2)
- /^(.+)$/ms
  - array(1)
- /^(.+)$/s
  - Array(1)
- ^ and $ match the start and end of a string by default
- The modifier **m** changes this behavior to match line start and end

eZ systems

6) Examples

Share your information

eZ systems

Share your information

6.1) Matching an URI (1)

- /^(\w+):\/\/(?:([a-z0-9_]+)(?::([^@]+))?@)?((?:[a-z\d-]+\.)+(?:[a-z]{2,6})|(?:(?:(?:\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5])\.){3}(?:\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5])))(?::([1-9]\d{1,3}|[1-5]\d{4}|6[1-4]\d{3}|65[1-4]\d\d|655[12]\d|6553[1-6]))?((?:\/[%\.a-z\d_~-]*)*)((?:[\?&](?:[a-z\d\.\[\]%-]+)(?:=[a-z\d\.\[\]%-]*)?)*)(?:#(\w+))?$/i

eZ systems

Share your information

6.1) Matching an URI (2)

- Protocol
  - $protocol = '(\\w+):\\/\\/';
- User
  - $user = '(?:([a-z0-9_]+)(?::([^@]+))?@)?';
- Host
  - $host = '(?:[a-z\\d-]+\.)+(?:[a-z]{2,6})';
- IP
  - $number = '(?:\\d|[1-9]\\d|1\\d\\d|2[0-4]\\d|25[0-5])';
  - $ip = "(?:(?:$number\\.){3}$number)";

eZ systems

Share your information

6.1) Matching an URI (3)

- Port
  - $port = '(?::([1-9]\d{1,3}|[1-5]\d{4}|6[1-4]\d{3}|65[1-4]\d\d|655[12]\d|6553[1-6]))?';
- Path
  - $path = '((?:\\/[%\\.a-z\\d_~-]*)*)';
- Querystring
  - $parameter = '((?:[\\?&](?:[a-z\d\\.\\[\\]%-]+)(?:=[a-z\\d\\.\\[\\]%-]*)?)*)';
- Anchor
  - $anchor = '(?:#(\\w+))?';

eZ systems

Share your information

6.1) Matching an URI (4)

- The complete expression
- $pattern = "/^$protocol$user($host|$ip)$port$path$parameter$anchor$/i";
- `/^(\w+):\/\/`
- `(?:([a-z0-9_]+)(?::([^@]+))?@)?`
- `(`
- `(?:[a-z\d-]+\.)+(?:[a-z]{2,6})`
- `|`
- `(?:(?:(?:\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5])\.){3}`
- `(?:\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5]))`
- `)`
- `(?::([1-9]\d{1,3}|[1-5]\d{4}|6[1-4]\d{3}|65[1-4]\d\d|655[12]\d|6553[1-6]))?`
- `((?:\/[%\.a-z\d_~-]*)*)`
- `((?:[\?&](?:[a-z\d\.\[\]%-]+)(?:=[a-z\d\.\[\]%-]*)?)*)`
- `(?:#(\w+))?$/i`

eZ systems

6.1) Matching an URI (5)

- Usage on an URI:
  - http://kore:password@kore-nordmann.de:80/administration/index.php?id=1&site=examples#ex1
- Match:
  - array(9) {
  - [0]=> string(88) "http://kore:password@kore-nordmann.de:80/administration/index.php?id=1&site=examples#ex1"
  - [1]=> string(4) "http"
  - [2]=> string(4) "kore"
  - [3]=> string(8) "password"
  - [4]=> string(16) "kore-nordmann.de"
  - [5]=> string(2) "80"
  - [6]=> string(25) "/administration/index.php"
  - [7]=> string(19) "?id=1&site=examples"
  - [8]=> string(3) "ex1"
  - }

Share your information

eZ systems

Share your information

Thank you

- I hope that I did not confuse you to much
- More resources:
  - http://php.net/pcre
  - http://kore-nordmann.de