Лекция 7

Регулярные выражения

Определение

- **Регулярные выражения** (англ. «regular expressions», жарг. «регэкспы» или «регексы») — современная система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска.
- Образец (англ. «pattern»), задающий правило поиска, по-русски также иногда называют «шаблоном», «маской», или на английский манер «паттерном».

Предназначение

Регулярные выражения используются некоторыми текстовыми редакторами и утилитами для поиска и подстановки текста. Например, при помощи регулярных выражений можно задать шаблоны, позволяющие:

- найти все последовательности символов *«кот»* в любом контексте, как то: *«кот»*, *«котлета»*, *«терракотовый»*;
- найти отдельно стоящее слово *«кот»* и заменить его на *«кошка»*;
- найти слово *«кот»*, которому предшествует слово *«персидский»* или *«чеширский»*;
- убрать из текста все предложения, в которых упоминается слово *кот* или *кошка*.

Распространённость

- Регулярные выражения (regex) являются важной составной частью текстовых редакторов, инструментов поиска и большинства основных языков программирования, которые поддерживают регулярные выражения для работы со строками.

Виды регулярных выражений

- POSIX (BRE, ERE)
- **PCRE** = **P**erl-**C**ompatible **R**egular **E**xpressions

Из чего состоят регэкспы

1. Символы
- обычные
- специальные (метасимволы)

2. Операции
- квантификация
- перечисление
- группировка

Разделители

- Разделителем может быть любой символ не являющийся буквой, цифрой, обратной косой чертой или каким-либо пробельным символом.
- Часто используемыми разделителями являются косые черты (/), знаки решетки (#) и тильды (~).

Примеры:

- /foo bar/
- #^[^0-9]$#
- +php+
- %[a-zA-Z0-9_-]%

PHP-синтаксис

```
$pattern = “/^foo/”;
int preg_match ( string $pattern, string $subject [, array &$matches ] )
```

- Выполняет проверку на соответствие строки **$subject** регулярному выражению **$pattern** и записывает результаты поиска в массив **$matches**.
- Возвращает количество найденных соответствий (в данном случае 0, если не найдены совпадения или 1), если поиск завершился успешно, и false, если возникли ошибки

PHP-синтаксис

```
$myRegexp = “/^foo/”;
int preg_match_all ( string $pattern,
string $subject [, array &$matches ] )
```

- Выполняет проверку на соответствие строки **$subject** регулярному выражению **$pattern** и записывает результаты поиска в массив **$matches**.
- Возвращает количество найденных соответствий (0, если не найдены), если поиск завершился успешно, и false, если возникли ошибки

PHP-синтаксис

```
mixed preg_replace ( mixed $pattern ,
mixed $replacement , mixed $subject [,
int $limit = -1 [, int &$count ]] )
```

- Выполняет поиск совпадений в строке **subject** с шаблоном **pattern** и заменяет их на **replacement**.
- Возвращает массив, если параметр subject является массивом, иначе возвращается строка

PHP-синтаксис

```
array preg_split ( string $pattern , string $subject [, int $limit = -1 [, int $flags = 0 ]] )
```

- Разбивает строку по регулярному выражению.
- Возвращает массив, состоящий из подстрок заданной строки **subject**, которая разбита по границам, соответствующим шаблону **pattern**.

МЕТАСИМВОЛЫ

Любой символ

- `$myRegexp = "/./";`
- `preg_match($myRegexp, 'foo'); // true`
- `preg_match($myRegexp, "\r\n"); // false`

Что на самом деле хотели получить?

- `preg_match("/./s", "\r\n"); // true`

Граница слова

- `preg_match("/\ba/", 'alabama'); // true`
- `preg_match("/a\b/", 'alabama'); // true`
- `preg_match("/a\b/", 'naïve'); // true`

Не-граница слова:

- `preg_match("/\Ba/", 'alabama'); // true`

СИМВОЛЬНЫЕ КЛАССЫ

Пробельные символы

- **/\s/ (инвертированный вариант /\S/)**

Следующие специальные символы являются пробельными:

PHP:

- **\t \n \v \f \r \u0020**

Буквы и цифры

- **/\d/** ~ цифры от 0 до 9
- **/\w/** ~ буквы, цифры и подчёркивание

В JS и PHP не работает для русских букв!

И наоборот:

- **/\D/** ~ всё, кроме цифр
- **/\W/** ~ всё, кроме букв и цифр

Произвольные классы символов

Пример:
/[abc123]/

Работают метасимволы и диапазоны:
/[A-F\d]/
Можно указать несколько диапазонов:
/[a-cG-M0-7]/

ВАЖНО: диапазоны берутся из Юникода. При работе с кириллическими диапазонами проверьте порядок символов в Юникоде!

Произвольные классы символов

Символ «точка» — просто точка!

```
preg_match('/[.]/', 'anything'); // false
```

Символы: \ ] -

- /[\\\]-]/

Инвертированные символьные классы

Всё, кроме a, b, c:

- `/[^abc]/`

^ как символ:

- `/[abc^]/`

КВАНТИФИКАТОРЫ

Ноль или более, один или более

- `preg_match('/bo*/', 'b'); // true`
- `preg_match('/.*/', '');  // true`
- `preg_match('/bo+/', 'b'); // false`

Ноль или один

- `preg_match('/colou?r/', 'color');`
- `preg_match('/colou?r/', 'colour');`

Диапазоны повторов

- **/bo{7}/** точно 7
- **/bo{2,5}/** от 2 до 5, x < y
- **/bo{5,}/** 5 или более

В JS и PHP не работает!

- `preg_match('/b{,5}/', 'bbbbb');`

Жадные (greedy) квантификаторы

```php
preg_match('/a+/', 'aaaaa', $matches);
print_r($matches);
// Array ( [0] => aaaaa )
```

Ленивые (lazy) квантификаторы

```
preg_match('/a+?/', 'aaaaa', $matches);
print_r($matches);
// Array ( [0] => a )

preg_match('/a*?/', 'aaaaa', $matches);
print_r($matches);
// Array ( [0] => )
```

Группировки

С захватом

- `preg_match("/(boo)/", "boo", $matches);`
- `// $matches = {"boo"};`

Без захвата

- `preg_match("/(?:boo)/", "boo", $matches);`
- `// $matches = {};`

Пример

- `<img alt="картинка" src="image.jpg"/>`
- `/<img(?:alt="(?:.*)")? src="(.+)" \/>/i`
- `$matches={'alt="картинка"', 'картинка','image.jpg'};`

Группировки и получаемый массив

```
preg_match('/(bo)o+(m)/', 'the booooom', $matches);
print_r($matches);
// Array ( [0] => booooom [1] => bo [2] => m )
```

Порядок нумерации группировок

- `/((foo) (b(a)r))/`

Порядок нумерации группировок

- `/((foo) (b(a)r))/`
- `/(                )/ // $matches[0] = foo bar`
- `/ (foo)            / // $matches[1] = foo`
- `/        (     ) / // $matches[2] = bar`
- `/          (a)   / // $matches[3] = a`

Перечисление

- `/red|green|blue light/`
- `/(red|green|blue) light/`
- `preg_match("/a(;|$)/", 'var a'); // true`

Backreferences (обратные ссылки)

- `preg_match("/(red|green) apple is \1/", 'red apple is red'); // true`
- `preg_match("/(red|green) apple is \1/", 'green apple is green'); // true`
- `preg_match("/(red|green) apple is \1/", 'green apple is red'); // false`

Представление символов

```
\x09 === \t (не Unicode, для ASCII/ANSI)
\u20AC === € (для Unicode)
```

Обратный slash убирает специальное значение у символа

- `preg_match('/\(\)/', '()');  // true`
- `preg_match('/\\n/',  '\\n'); // true`

Иногда верно и обратное

- `preg_match('/\f/', 'f') // false!`

Флаги (модификаторы) в регулярных выражениях

- i m s u
- global match
- ignore case
- multiline matching for ^ and $
- utf-8 (не совместим с PCRE, есть только в PHP)
- string as single line

Пример:

- `preg_match('/hello/i', 'HeLlO'); // true`

Regexp Injection

```
$userInput = '[abc]';
// ПЛОХО
preg_match($pattern, $userInput);
// ХОРОШО
preg_match($pattern, preg_quote($userInput));
```

Regexp Injection

string **preg_quote** (string $str)

- Функция **preg_quote()** принимает строку *str* и добавляет обратный слэш перед каждым служебным символом. Это бывает полезно, если в составлении шаблона участвуют строковые переменные, значение которых в процессе работы скрипта может меняться.
- В регулярных выражениях служебными считаются следующие символы: `. \ + * ? [ ^ ] $ ( ) { } = ! < > | : -`

Ссылки

- http://pcre.ru - регулярные выражения, примеры, документация, шаблоны.
- http://tech.yandex.ru/education/shri/simf-2013/talks/712/ - Максим Ширшин, Регулярные Выражения (Школа Разработки Интерфейсов Яндекса).
- http://rubular.com – Ruby Regular Expression Editor. Тестирование регулярных выражений.
- http://uzer.com.ua/cross/ - кроссворд по регулярным выражениям.
- http://www.bitcetera.com/page_attachments/0000/0030/regex_in_a_nutshell.pdf - шпаргалка
- http://regex101.com/ - онлайн редактор RegEx
- http://regexr.com/ - онлайн редактор RegEx

Лабораторная работа

Сверстать форму регистрации пользователя на абстрактном сайте. Добавить в форму следующие поля: имя, фамилия, пароль, подтверждение пароля, электронная почта, личный сайт, телефон. Каждое поле нужно проверить на соответствие регулярному выражению:

1) Имя, фамилия - должны состоять не менее из 3 символов русской кириллицы.
2) Пароль - должен состоять не менее из 8 символов, содержать хотя бы одну латинскую букву в верхнем регистре, хотя бы одну в нижнем и хотя бы одну цифру
3) Электронная почта - должна соответствовать формату. За основу взять стандартный формат e-mail: user@domain.zone. Обязательное поле.

Лабораторная работа

++

Необязательные поля (скрипт должен работать корректно в случае их отсутствия). Введенные данные должны быть валидны.

1) Личный сайт - должен соответствовать формату URI.
2) Телефон - должен соответствовать международному формату записи телефонных номеров: +7 (495) 111 2233.
3) Дата рождения в формате (ДД-ММ-ГГГГ)
4) IP адрес (вводится ручную)

+++

Распарсить URI (http://4pda.ru/forum/index.php?showforum=200)

- Домен (4pda)
- Зона (ru)
- Текущая страница/скрипт (index.php)
- GET-запрос (showforum=200)