# APLIKACJE INTERNETOWE
## PHP

**TEMAT 5-05**: Wyrażenia regularne w PHP.

| Autor dokumentu: Wojciech Galiński | środa, 7 września 2016 r. | 351203 Technik informatyk |
|---|---|---|

ŹRÓDŁA WIEDZY: http://www.php.net/manual/pl, http://webmaster.helion.pl/index.php/kurs-php, http://pl.wikipedia.org/, http://pl.wikibooks.org/wiki/PHP, http://phpkurs.pl/, http://kursphp.com/, http://www.gajdaw.pl/php/wyrazenia-regularne-pcre-php-tutorial/index.html.

# Zagadnienia obowiązkowe

1. **Wyrażenia regularne** – (ang. regular expressions, regexp) służą do określania, czy podany ciąg znaków pasuje do wzorca oraz wyszukują w tekście wystąpienia wzorca.

   Korzystanie z wyrażeń regularnych polega na umiejętności tworzenia wzorców do dopasowywania lub do wyszukania w zadanym tekście.

   W PHP obsługiwane są wyrażenia zgodne z **POSIX** (nazwy zaczynające się od: **ereg**_) oraz wyrażenia regularne **Perla** (nazwy zaczynające się od: **preg**_). Zalecane jest stosowanie tych drugich, ponieważ mają większe możliwości i działają szybciej (źródło: Wikipedia).

2. **Wzorzec wyrażenia regularnego** – to ciąg znaków pomiędzy dwoma ogranicznikami:

   **`/wyrażenie/opcje`**

   W powyższym wzorcu ogranicznikami są ukośniki (ang. slash), ale mogą to być także inne znaki, np. |wyrażenie|opcje, *wyrażenie*opcje, #wyrażenie#opcje, [wyrażenie]opcje, {wyrażenie}opcje.

3. ***Modyfikatory (opcje) wyrażenia*** – wyróżniamy:
   - ➔ **`i`** – ignorowanie wielkości znaków;
   - ➔ **`m`** – d tekst podzielony na wiele linii;
   - ➔ **`s`** – znak „**`.`**” oznacza wtedy dowolny znak włącznie ze znakiem "**`\n`**";
   - ➔ **`x`** – ignorowanie białych znaków i włączenie komentarzy w stylu BASH (#);
   - ➔ **`U`** – zamienia kwantyfikatory zachłanne z leniwymi;
   - ➔ **`u`** – włączenie kodowania UTF-8 .

   Więcej na temat opcji: http://php.net/manual/pl/reference.pcre.pattern.modifiers.php.

4. **Testowanie wyrażeń regularnych w PHP** – najprościej użyć do tego funkcji „preg_match”, np.

```
$wyr = '/zsipo/i';   $tekst = "ZSiPO w Nysie";
echo "Wzorzec ". (preg_match($wyr, $tekst)? "": "nie "). "znaleziony.";
```

5. **Struktura wyrażenia regularnego** – może ono składać się z następujących kategorii symboli:
   - ➔ ***`kotwice`*** – znaki: **`^`** (początek), **`$`** (koniec ciągu z ewentualnym znakiem „**`\n`**”), np. `/^Ot/`, `/.txt$/`, `/^Ala.+kota$/`;
   - ➔ ***kwantyfikatory zasięgu*** – umieszczane są po znaku lub klasie dozwolonych znaków. Wyróżniamy:
     - ✔ dostępne są następujące kwantyfikatory (bez nawrotów – pochłania wszystkie znaki, zachłanny / leniwy – pochłania jak największą / najmniejszą ilość napisu):

| Kwantyfikator zachłanny | Kwantyfikator leniwy | Kwantyfikator bez nawrotów | Opis |
|---|---|---|---|
| **`*`** | **`*?`** | **`*+`** | 0 lub więcej wystąpień. |
| **`+`** | **`+?`** | **`++`** | 1 lub więcej wystąpień. |
| **`?`** | **`??`** | **`?+`** | 0 lub 1 wystąpienie. |
| **`{N}`** | | **`{N}+`** | Dokładnie N wystąpień. |
| **`{N,}`** | **`{N,}?`** | **`{N,}+`** | Co najmniej N wystąpień. |
| **`{M,N}`** | **`{M,N}?`** | **`{M,N}+`** | Od M do N wystąpień. |

     - ✔ **`{długość}`** – dokładne określenie dozwolonej długości;
     - ✔ **`{długość_min, długość_max}`** – określenie przedziału dozwolonych ilości znaków;
     - ✔ **`{długość_min,}`**, **`{,długość_max}`** – określenie minimalnej / maksymalnej długości.

PRZYKŁADY: „`/.{3}/`”, „`/preg_[a-z]{4}_[a-z]+/i`”, „`/[a-zA-Z0-9_]{6,}/`”.

➔ ***klasy znaków*** – listy dozwolonych znaków występujących dokładnie jeden raz na danej pozycji. Obowiązują je następujące reguły:

- ✔ możemy wymienić wszystkie dozwolone znaki w nawiasach kwadratowych, np. **`[abcd]`**;
- ✔ możemy określić przedział znaków, np. **`[a-d]`**; (równoznaczne z zapisem: **`[abcd]`**);
- ✔ powyższe reguły mogą być stosowane razem, np. **`[a-dA-DxY]`**;
- ✔ znak „**^**” bezpośrednio po otwarciu nawiasu kwadratowego oznacza dopełnienie zbioru (negację), np. **`[^0-9]`** oznacza wszystkie znaki nie będące cyframi dziesiętnymi;
- ✔ metaznaki, czyli znaki o specjalnym znaczeniu musi poprzedzać znak „\” (ang. backslash – odwrotny ukośnik), np. **`[\.]`** – do znaków specjalnych zaliczamy m. in.: '**`\\`**' , '**`\.`**', '**`\[`**', '**`\{`**', '**`\(`**', '**`\)`**', '**`\|`**', '**`\^`**', '**`\$`**', '**`\*`**', '**`\+`**', '**`\?`**'.
- ✔ **`\n \t \r \cX`** – znaki specjalne: koniec wiersza, tabulator, powrót karetki, klawisz specjalny (**X** – oznacza znak z klawiatury, np. **cC** oznacza [Ctrl]+[C]);
- ✔ **`\Q \E`** – wyłączanie / włączanie interpretacji znaków specjalnych, np. `/\Q\n\E\n/`;
- ✔ **`\nn \xnn`** – kody znaków w postaciach: ósemkowej i szesnastkowej, np. \101, \x41 (patrz też: http://php.net/manual/pl/regexp.reference.escape.php);
- ✔ **`.`** oznacza dowolny znak (za wyjątkiem "**`\n`**", chyba, że włączona jest opcja „s”);
- ✔ predefiniowane klasy znaków:
  **`\w`** – to samo, co: **`[a-zA-Z0-9_]`**; **`\W`** – to samo, co: **`[^a-zA-Z0-9_]`**;
  **`\d`** – to samo, co: **`[0-9]`**; **`\D`** – to samo, co: **`[^0-9]`**;
  **`\s`** – to samo, co: **`[ \t\r\n]`**; **`\S`** – to samo, co: **`[^ \t\r\n]`**.
- ✔ predefiniowane klasy znaków według standardu POSIX zdefiniowane są w następujący sposób:
  „**`[:digit:]`**” („**`\d`**”, cyfry dziesiętne), „**`[:cntrl:]`**” (znaki kontrolne o kodach: 0-31),
  „**`[:xdigit:]`**” (cyfry heksadecymalne), „**`[:blank:]`**” (spacja, lub tabulator),
  „**`[:alnum:]`**” (litery lub cyfry), „**`[:graph:]`**” (znaki graficzne, bez spacji),
  „**`[:alpha:]`**” (litery), „**`[:print:]`**” (znaki drukowalne, ze spacją),
  „**`[:lower:]`**” (małe litery łacińskie), „**`[:space:]`**” (spacja),
  „**`[:upper:]`**” (duże litery łacińskie), „**`[:punct:]`**” (znaki drukowalne, bez spacji i cyfr),
  „**`[:ascii:]`**” (znaki o kodach: 0-127), „**`[:word:]`**” („**`\w`**”, znaki 2-bajtowe).
  PRZYKŁADY: `/[A-H][[:digit:]]/i`, `/[[a-f][:digit:]]/i`, `/[[:space:]]/`.

➔ ***grupy znaków*** – to podciągi rozdzielone znakiem „|” (mogą być zagnieżdżane). W celu zmiany kolejności przetwarzania wyrażenia można umieścić je w nawiasach okrągłych (ostatni przykład pokazuje, jak uniknąć przechwytywania – patrz: następny temat), np.
`/so|ni/`, `/(.a|a.)/`, `/(wiosna|lato|jesień|zima)/i`, `/N(?:A|IE)/`;

➔ ***komentarze*** – mają następującą budowę: „**`(?# komentarz)`**”.

6. **Testowanie wyrażeń w Internecie** – w Internecie można testować wyrażenia regularne. Oto kilka adresów: http://regexp.pl/, http://www.regextester.com/, http://www.regexplanet.com/.

## Zadania

1. Zapoznaj się z przedstawionymi przykładami. Wykonaj je i przeanalizuj ich działanie.
2. Wyjaśnij, podaj przykładowy tekst pasujący do podanych wzorców (znajdź błędy w składni):
   ➔ **`/./s`** ➔ **`/[^0-9]{3,}/`** ➔ **`/(ja|ty|on|ona|ono) jest/`**
   ➔ **`*a+psik*i`** ➔ **`/pod[a-zA-Z]*/`** ➔ **`/[0-9a-f]{1,8}/i`**
   ➔ **`/[aeiou]/`** ➔ **`/(tak|nie)/i`** ➔ **`/\d{2}-[:digit:]{3}/m`**
3. Wymyśl 3 wyrażenia regularne korzystające min. z 5 elementów wymienionych w temacie.
4. Znajdź w Internecie 1 ważne zastosowanie wyrażenia i przeanalizuj je (np. walidacja adresu IP).