

Biểu thức chính quy RegExp

§1) Cơ bản về RegExp

§2) Tool kiểm tra RegExp

§3) (PHP) Lưu ý sử dụng RegExp

§4) (PHP) Hàm preg_mach()

§5) (PHP) Hàm preg_replace()

§6) (PHP) Hàm preg_split()

§7) (PHP) Hàm preg_grep()

§8) (PHP) Hàm preg_quote()

§9) (PHP) Hàm preg-replace-callback()

LẬP TRÌNH PHP

PSR SPL

PHP Cơ bản

LẬP TRÌNH ỨNG DỤNG IOS - SWIFT

LẬP TRÌNH DART - FLUTTER

LẬP TRÌNH C# (C SHARP)

Lập trình C# Cơ bản

XENFORO

ZEND FRAMEWORK

Expressive

SERVER

MySql Server Apache PHP

HTML

JAVASCRIPT

JQuery TypeScript - Angular

CSS

Sử dụng SASS / SCSS

Bootstrap - CSS Framework

Biểu thức chính quy RegExp

Cách viết biểu thức chính quy RegExp, sử dụng biểu thức chính quy trong lập trình

- Biểu thức chính quy
- Biểu thức cơ bản
- Các ký tự mô tả
- Ký hiệu viết tắt cho tập hợp
- Cờ thiết lập

Biểu thức chính quy - Regular expression

Biểu thức chính quy là gì?

Biểu thức chính quy là một nhóm các ký tự, ký hiệu nó được sử dụng để tìm kiếm văn bản (text).

Một biểu thức chính quy là một mẫu nó tương đồng quy luật với một chuỗi từ trái qua phải. Biểu thức chính quy tên tiếng anh là **Regular Expression** gọi tắt là **regex** hoặc **regexp**

Trong lập trình nó được dùng với các hàm xử lý chuỗi, xử lý văn bản với các tác vụ cụ thể như: tìm và thay thế chuỗi, kiểm tra tính hợp lệ của dữ liệu, trích xuất chuỗi con từ một chuỗi ... Hầu hết các ngôn ngữ lập trình đều hỗ trợ **Regex** như **PHP**, **C#**, **JAVA** ... Thậm chí RegExp còn rất phổ biến trong các ứng dụng, công cụ khác nhau như rewrite URL **mod_rewrite**, tìm kiếm và thay thế trong các IDE

Thử xem xét một ví dụ về Regex - Giả sử bạn ứng dụng của bạn yêu cầu người dùng phải tuân thủ quy tắc đặt tên: (1) Tên được phép chứa các ký tự, các số, gạch dưới, gạch nối. (2) Tên phải có độ dài trong khoảng cho phép từ 3 đến 15 ký tự.. Thì biểu thức chính quy biểu diễn quy tắc đó sẽ như sau:

^[a-z0-9_-]{3,15}\$

- **^** Ký hiệu cho biết bắt đầu một dòng
- **[a-z0-9_-]** Cho phép tên chứa ký tự a-z, số từ 0 - 9, ký tự -, ký tự _
- **{3,15}** Tên dài 3 đến 15 ký tự
- **\$** Điểm kết thúc dòng

Với biểu thức chính quy trên thì các tên như **xuanthu**, **xuan-xhu**, **xuanxhu_123** ... được chấp nhận vì dài trong khoảng 3 - 15, chữ đều viết thường.

Biểu thức RegExp cơ bản

Một biểu thức Regex chỉ là một mẫu các ký tự dùng để tìm kiếm trong text (chuỗi). Ví dụ một biểu thức **uóc** có nghĩa phù hợp với nó là bắt đầu bằng **u** theo sau là **ó** và tiếp theo là **c**, mang biểu thức so với text thì thấy hợp mẫu như sau:

uóc => Uóc một điều ... mong **uóc** rất đơn sơ. Nụ hôn trao hạnh phúc đến bất r
(Chạy thử **vidu01**)

Biểu thức chính quy RegExp là phân biệt chữ hoa chữ thường. Có cơ chế bật cờ thiết lập không phân biệt chữ hoa, chữ thường ở phần dưới

SQL

SQL Server (.NET Framework - C#)

MS Access

JAVA

Android
Java

THUẬT NGỮ - CÁC VẤN ĐỀ CƠ BẢN

TOOLS

Kubernetes
Git và GitHub

Mathematica

SSH - Secure Shell
Grunt

Elasticsearch
Docker

macOS
English Study

TIN TỨC CÔNG NGHỆ

Các ký tự biểu diễn - Meta

Các ký hiệu biểu diễn thông tin tóm tắt lại và giải thích chi tiết từng mục ở phần sau

Ký tự Meta	Mô tả
.	Biểu diễn bất kỳ ký tự nào ngoài trừ ký tự xuống dòng
[]	Tập hợp ký tự. Phù hợp nếu có bất kỳ ký tự nào trong dấu []
[^]	Tập hợp ký tự phủ định. Phù hợp nếu không có ký tự nào trong []
*	Lặp lại 0 đến nhiều lần.
+	Lặp lại 1 hoặc nhiều lần
?	Tùy chọn có hay không cho mẫu phía trước
{n,m}	Độ dài tối thiểu là n tối đa là m
(xyz)	Biểu diễn một nhóm.
	Biểu diễn thay thế, phép toán or
\	Biểu diễn ký tự đặc biệt [] () { } . * + ? ^ \$ \
^	Điểm bắt đầu của dòng.
\$	Điểm kết thúc của dòng

Ký hiệu chấm .

Ký hiệu dấu chấm . là một meta đơn giản, nó biểu diễn bất kỳ ký tự nào ngoài trừ ký tự `return \r` hoặc `newline \n`. Ví dụ biểu thức `.oàn` thì có nghĩa là: một ký tự nào đó, tiếp theo đến ký tự `o`, tiếp theo đến `à` cuối cùng là `n`. Ví dụ dùng mẫu đó tìm trong chuỗi.

```
.oàn => Sự hoàn hảo dường như không thể đạt được, nhưng nếu chúng ta theo đuổi thì chúng ta sẽ chạm đến sự xuất sắc.  
(Chạy thử vidu02)
```

Tập hợp ký tự []

Dùng [] để chứa tập hợp các ký tự. Có thể dùng dấu- để biểu diễn một dải các ký tự theo vị trí trong bảng chữ cái như `a-z`, `0-9` ..., biểu thức so sánh sẽ hợp mẫu nếu chứa bất kỳ ký tự nào trong đó (không cần quan tâm thứ tự)

Ví dụ biểu thức `[uƯ]óc` có nghĩa là: Có một chữ `u` hoặc `Ư`, theo sau bởi `ó`, tiếp theo là `c`

```
[uƯ]óc => Ưóc một điều ... mong ước rất đơn sơ. Nụ hôn trao hạnh phúc đến bà  
(Chạy thử vidu03)
```

Nếu [] chứa . thì nó biểu diễn ký tự . chứ không con ý nghĩa đại diện như trường hợp trên.

```
nh[.] => Thời gian cứ thế xoay vòng thật nhanh. Bao mùa chiếc áo phông phanh
Chạy thử vidu04
```

Tập hợp ngoại trừ [^]

Thông thường thì ^ biểu diễn điểm bắt đầu của chuỗi, tuy nhiên nếu nó nằm ở vị trí sau dấu [của cặp [] thì nó lại mang ý nghĩa tạo ra tập hợp ký tự loại trừ (phụ định). Ví dụ biểu thức [^n]hanh có nghĩa là bất kỳ ký tự nào ngoại trừ ký tự n, theo sau bởi h, tiếp theo bởi a, n và h

```
[^n]hanh => Thời gian cứ thế xoay vòng thật nhanh. Bao mùa chiếc áo phông ph
Chạy thử vidu05
```

Lặp lại với ký tự *

Ký hiệu * cho biết có sự lặp lại 0 hoặc nhiều lần mẫu phù hợp đứng phía trước nó. Ví dụ mẫu a* có nghĩa là ký tự a lặp lại 0 hoặc nhiều lần là phù hợp. Nếu nó đi sau tập hợp thì lặp tập hợp đó lặp lại 0 hoặc nhiều lần. ví dụ [a-z]* có nghĩa là dòng có số lượng bất kỳ các ký tự chữ viết thường thì phù hợp.

* có thể sử dụng với . để biểu diễn bất kỳ chuỗi nào, hay dùng mẫu (.*)

* có thể sử dụng với ký tự trắng \s để biểu diễn bất kỳ khoảng trắng nào.

Ví dụ \s*mình\s* có nghĩa bắt đầu bởi không hoặc nhiều khoảng trắng, tiếp theo là ký tự m, i, n, h tiếp theo là không hoặc nhiều khoảng trắng.

```
"\s*mình\s*" => Dùng so sánh mình với bất cứ ai trong thế giới này.
Nếu bạn làm như vậy có nghĩa bạn đang sỉ nhục chính bản thân mình. Bill Gate
Chạy thử vidu06
```

Lặp lại với ký tự +

Ký hiệu + tương tự như * nhưng lặp lại 1 hoặc nhiều. Ví dụ

có.+:! có nghĩa ký tự bắt đầu bằng có theo sau ít nhất một ký tự nào đó, tiếp theo là ký tự !.

```
có.+:! => Đàn ông cần tiền chủ yếu chỉ để cho hai việc: có được nàng và thoát
Chạy thử vidu07
```

Mẫu phía trước có hay không đều được với ?

Trong biểu thức Regex thông thường ? là một tùy chọn cho biết mẫu phía trước nó có thể có hoặc không. Ví dụ [h]?ôn nghĩa là tùy chọn có h hoặc không, theo sau là ô, tiếp theo là n

```
[h]?ôn => Đàn bà khôn ngoan hơn đàn ông vì họ biết ít hơn, nhưng hiểu nhiều
Chạy thử vidu08
```

Biểu diễn độ dài { }

{ } là biểu diễn số lượng, nó chỉ ra số lần mà một ký tự hoặc một nhóm các ký tự lặp lại. Ví dụ [0-9]{2,3} có nghĩa là có tối thiểu 2 tới 3 ký tự số.

Bạn có thể bỏ đi số thứ 2, ví dụ [0-9]{2,} có nghĩa là chuỗi có 2 hoặc nhiều ký tự số. Nếu bỏ đi ký tự , ví dụ [0-9]{3} có nghĩa là chuỗi chính xác có 3 ký tự.

Nhóm mẫu (. . .)

Nhóm ký tự là một mẫu (pattern) con được viết biên trong (). Ví dụ (ab)* lặp lại ab 0 hoặc nhiều lần. Chúng ta cũng dùng ký hiệu | bên trong nhóm như là phép toán or để xác định nhóm. Ví dụ n(g|h) có nghĩa bắt đầu bằng n theo sau là một mẫu, mẫu đó hoặc là chữ g hoặc là chữ h

```
n(g|h) =>Nếu có một ai đó làm chậm bước chân của bạn, hãy nhẹ nhàng rẽ sang
Chạy thử vidu09
```

Biểu diễn thay thế |

Xem ví dụ trên

Biểu diễn ký tự đặc biệt với \

Do một số ký hiệu đã được dùng đã biểu diễn Regex như :{ } [] / \ + * . \$ ^ | ? nên để biểu diễn các ký tự đó dùng ký hiệu \ trước ký tự.

"(f|c|m)at\." => The fat cat sat on the mat.

Bắt đầu của dòng ^

Sử dụng ^ để cho biết sẽ kiểm tra sự phù hợp nếu ký tự đầu tiên của chuỗi hợp mẫu. Ví dụ ^a thì chuỗi phù hợp có dạng như abcxyz, nếu vẫn chuỗi đó nó lại không phù hợp với ^b.

^(T|t)he có nghĩa là T hoặc t bắt đầu của chuỗi, theo sau là he

Điểm kết thúc của chuỗi \$

Cho biết kết thúc dòng phải thỏa mãn mẫu phía trước\$

Ngược lại với ^ ví dụ (at\.)\$ nghĩa là cuối chuỗi có at. thì là phù hợp.

"(at\.)\$" => The fat cat. sat. on the mat.

Ký hiệu tắt cho tập hợp

Viết tắt	Diễn tả
.	Bất kỳ ký tự nào ngoại trừ xuống dòng
\w	Chữ,sô, và _, tương đương với: [a-zA-Z0-9_]
\W	Ngoài bảng chữ cái, tương đương với: [^\w]
\d	Các số: [0-9]
\D	Không phải số: [^\d]
\s	Là ký tự trắng, tương đương với: [\t\n\f\r\p{Z}]
\S	Không phải ký tự trắng: [^\s]

Biểu thức ?= lookahead

Lookahead ?= cho thêm vào để lọc kết quả.

Ký hiệu ?=. Phần đầu của biểu thức phải được tiếp nối bởi biểu thức lookahead.

Ví dụ `(T|t)he(=?\sfat)` thì `lookahead` là `(=?\sfat)` - nghĩa là `T` hoặc `t` theo sau là `he` vậy tìm được 2 kết quả. Nhưng do có biểu thức `lookahead`, điều này thì kết quả phù hợp là chỉ lấy khi theo sau nó là chuỗi `fat`

```
(T|t)he => The fat cat sat on the mat.  
Chạy thử vidu10
```

```
(T|t)he(=?\sfat) => The fat cat sat on the mat.  
Chạy thử vidu11
```

Biểu thức `?!` phủ định lookahead

Ký hiệu là `?!`, nghĩa là lấy kết quả mà đi sau nó không có chuỗi `lookahead`

```
(T|t)he(?!\sfat) => The fat cat sat on the mat.  
Chạy thử vidu12
```

Biểu thức `(?<=...)` Lookbehind

Sử dụng để lấy các phù hợp mà đi trước là một mẫu cũ thể. `(?<=(T|t)he\s)(fat|mat)` có nghĩa lấy tất cả các từ `fat` hoặc `mat` sau các từ `The` hoặc `the`

```
(?<=(T|t)he\s)(fat|mat) => The fat cat sat on the mat.  
Chạy thử vidu13
```

Biểu thức `(?<!...)` phủ định Lookbehind

Sử dụng để lấy các phù hợp mà đi trước không có một mẫu lookbehind chỉ ra.

```
(?<!(T|t)he\s)(cat) => The cat sat on cat.  
Chạy thử vidu14
```

Các cờ

Cờ	Diễn tả
i	Thiết lập không phân biệt chữ hoa chữ thường
g	Tìm kiếm toàn chuỗi.
m	Multiline: Anchor meta character works on each line.

Các cờ này được đưa vào mẫu theo dạng `/RegExp/flags`

`"/The/gi" => The fat cat sat on the mat.`

`"/.(at)/" => The fat cat sat on the mat.`

`"/.(at)/g" => The fat cat sat on the mat.`

```
"/.at(.)?$ /gm" => The fat  
cat sat  
on the mat.
```

Trong PHP các hàm sử dụng RegExp thường các biểu thức này nằm trong cặp `/.../`

Tác giả: <https://twitter.com/ziishaned>

Đăng ký theo dõi ủng hộ kênh



XuanThuLab



Thích 7

Chia sẻ 7



1 bình luận

Sắp xếp theo Mới nhất

Thêm bình luận...



Trần Khiêm

biểu thức chính quy cho ngày vd nhỏ hơn 32 thì sao

Thích · Phản hồi · 25 tuần



Nguyễn Chí Thiện

<https://www.regextester.com/96683>

Thích · Phản hồi · 16 tuần

Plugin bình luận trên Facebook

- » Nguyên lý lập trình IoC Inversion of control (IoC)
- » Tìm hiểu mô hình Adapter Pattern trong lập trình
- » DI - Dependency Injection trong PHP
- » Unix timestamp chuyển đổi thời gian Unix
- » AOP Lập trình hướng khía cạnh
- » Observer pattern - Mô hình lập trình Chủ thể - Quan sát
- » Mô hình lập trình Service Locator

[Tool kiểm tra RegExp >](#)

[Giới thiệu](#)

[Privacy](#)

[Lịch vạn niên](#)

[Liên hệ](#)

[RSS](#)

Đây là blog cá nhân, tôi ghi chép và chia sẻ những gì tôi học được ở đây về kiến thức lập trình PHP, Java, JavaScript, Android, C# ... và các kiến thức công nghệ khác

Developed by [XuanThuLab](#)

DMCA PROTECTED

Mục lục bài viết