

# [實用] 用 Regular Expression 做字串比對

23 Jun 2016

## 什麼是 Regular Expression

Regular Expression 中文翻成正規表示式

英文簡寫為 **Regex** 或 **RegExp**

**RegExp** 是用來比對字串是不是有符合正確的格式

語法很簡單而且大部分語言都有支援它

## 使用時機

譬如說你需要在程式內請使用者輸入生日

你規定的格式

- 1996-08-06

但使用者可能會不小心輸成其他格式

- 19960806
- 1996-8-6
- 85-08-06
- 850806

如果要寫程式來過濾這些格式恐怕要寫很多判斷式

這時候就可以用 **Regex** 來驗證字串格式

## 語法

### 字元

a: a 這個字元

0: 0 這個字元

.: 任意字元

RegExp	說明	範例
/a/	含有字母 a 的字串	"a", "apple"
./	含有任意字元的字串	"aaa", "a", "嗨"
/a./	含有字母 a 後面接一個任意字元的字串	"aaa", "apple"

### 次數

\*: 出現 0 次以上

?: 出現 0 次或 1 次

+: 出現一次以上

- {2}：出現兩次
- {2,}：出現兩次以上
- {,10}：出現十次以下
- {2,5}：出現兩次到五次

RegExp	說明	範例
/a*/	包含 0 次以上的 a	"apple","hello"
/ab*/	包含一個 a，後面至少 0 個 b	"a","ab"
/a?/	包含空字串或一個 a	"c","", "app"
/123a+/	包含 123 後面有一個以上的 a	"123a","123app"
/123a{1,2}/	包含 123 後面出現一個或兩個 a	"1123a","123aaa"

### 頭尾

- ^：開頭
- \$：結尾

RegExp	說明	範例
/^app/	開頭是 app 的字串	"app","apple"
/ry\$/	結尾是 ry 的字串	"Larry"
/^abcd\$/	開頭結尾中間只有 abcd 的字串	"abcd"
/^La.*le\$/	開頭是 La 尾巴是 le 的字串	"Larry loves apple"

### 比對多個字元

- []：括號內的任何字元
- [^]：不在括號內的任何字元

RegExp	說明	範例
/^[aeiou]/	開頭是小寫母音的字串	"apple","oh"
/[^aeiouAEIOU]\$/	結尾不是母音的字串	"Larry","ok"
/^[aeiou]{3}\$/	三個小寫母音組成的字串	"aaa","aeu"
/^[^aeiou]*\$/	不包含小寫母音的字串	"hEllO","App1E"
/[0-9]/	含數字的字串	"app1e","123"
/[a-z]/	含小寫字母的字串	"12a45","aaa"
/^[^a-zA-Z]\$/	不含英文字母的字串	"123","345"

### 特殊字元

`\.`: "." 這個字元，直接寫 `/./` 會被判斷成任意字元  
`\+`: "+" 這個字元，類似的還有 `\?`, `\*`  
`\(`: "(" 這個字元，類似的還有 `\)`, `\[`, `\]`  
`\\`: "\" 這個字元

`\d`: 任何數字字元，等同 `[0-9]`  
`\D`: 任何非數字字元，等同 `[^0-9]`  
`\w`: 任何數字字母底線，等同 `[A-Za-z0-9_]`  
`\W`: 任何非數字字母底線，等同 `[^A-Za-z0-9_]`  
`\s`: 任何空白字元 (空白, 換行, tab)，等同 `[ \f\n\r\t\v]`  
`\S`: 任何非空白字元 (空白, 換行, tab)，等同 `[^ \f\n\r\t\v]`

實例

RegExp	說明	範例
<code>/^\d{4}-\d{2}-\d{2}\$/</code>	西元生日格式	"1996-08-06"
<code>/^[A-Z]\d{9}\$/</code>	身分證字號	"A123456789"
<code>/^09\d{8}\$/</code>	手機號碼	"0912345678"
<code>/^[^aeiou]*\$/</code>	不包含小寫母音的字串	"hEllO", "AppLe"
<code>/^.*@gmail\.com\$/</code>	gmail 信箱	"test@gmail.com"
<code>/^[0-9+\-\\*\/]*\$/</code>	四則運算算式	"1+2*3"

在各語言中使用

C++

註: C++ 11 才開始支援 regex

```
#include<regex>
#include<string>
using namespace std;

int main() {
    regex pattern("^([0-9]{4}-[0-9]{2}-[0-9]{2})$");
    string str = "1996-08-06";

    if(regex_match(str, pattern)){
        // doSomething
    }
}
```

Java

```
public class Example {
    public static void main(String[] args) {
        String pattern = "^([0-9]{4}-[0-9]{2}-[0-9]{2})$";
```

```
String str = "1996-08-06";

if(str.matches(pattern)){
    //doSomething
}
}
```

## JavaScript

```
var pattern = new RegExp('[0-9]{4}-[0-9]{2}-[0-9]{2}');
var str = '1996-08-06';

if(str.match(pattern)){
    // doSomething
}
```

## 結論

Regular Expression 是個很好用的工具 剛學可能會覺得有點複雜  
用久了之後就會覺得很好用  
而且也可以搭配 **sed** 跟 **grep** 做取代、搜尋的工作  
可以省下不少時間

這篇只介紹比較常用的一些表示法  
如果想知道更深入的可以到網路上去查