Contents

PHP Preg_match Examples

Use preg_match() to match strings with regular expressions. Check the return value for true to see if the expression did match.

1. Syntax of preg_match

While full syntax is

```
int preg_match ( string $pattern , string $subject
     [, array &$matches [, int $flags = 0 [, int $offset = 0 ]]] )
```

you propably will use preg_match() mostly with two parameters for simply matching checks or with three to extract matches. You probably won't use the 4th and 5th parameter which can be used to return match offsets and limit matching to a given offset in the string.

2. Simple String Checks

Here are some syntax examples that check strings for certain content: **Basic matching**

```
preg_match("/PHP/", "PHP")       # Match for an unbound literal
preg_match("/^PHP/", "PHP")      # Match literal at start of string
preg_match("/PHP$/", "PHP")      # Match literal at end of string
preg_match("/^PHP$/", "PHP")     # Match for exact string content
preg_match("/^$/", "")           # Match empty string
```

Using different regex delimiters

```
preg_match("/PHP/", "PHP")                # / as commonly used delimiter
preg_match("@PHP@", "PHP")                # @ as delimiter
preg_match("!PHP!", "PHP")                # ! as delimiter
```

Changing the delimiter becomes useful in some cases

```
preg_match("/http:\/\//", "http://");     # match http:// protocol prefix with / del
preg_match("#http://#",   "http://")      # match http:// protocol prefix with # del
```

Case sensitivity

```
preg_match("/PHP/", "PHP")                # case sensitive string matching
preg_match("/php/i", "PHP")               # case in-sensitive string matching
```

Matching with wildcards

```
preg_match("/P.P/",     "PHP")            # match a single character
preg_match("/P.*P/",    "PHP")            # match multipe characters
preg_match("/P[A-Z]P/", "PHP")            # match from character range A-Z
preg_match("/[PH]*/",   "PHP")            # match from character set P and H
preg_match("/P\wP/",    "PHP")            # match one word character
preg_match("/\bPHP\b/", "regex in PHP")   # match the word "PHP", but not "PHP" as l
```

Using quantifiers

```
preg_match("/[PH]{3}/",   "PHP")          # match exactly 3 characters from set [PH]
preg_match("/[PH]{3,3}/", "PHP")          # match exactly 3 characters from set [PH]
preg_match("/[PH]{,3}/",  "PHP")          # match at most 3 characters from set [PH]
preg_match("/[PH]{3,}/",  "PHP")          # match at least 3 characters from set [PH
```

3. Extracting Data

To extract data using regular expression we have to use capture/grouping syntax.

Some basic examples

```
# Extract everything after the literal "START"
preg_match("/START(.*)/", $string, $results)

# Extract the number from a date string
preg_match("/(\d{4})-(\d{2})-(\d{2})/", "2012-10-20", $results)

# Nesting of capture groups, extract full name, and both parts...
preg_match("/name is ((\w+), (\w+))/", "name is Doe, John", $results)
```

So you basically just enclose the sub patterns you want to extract with braces and fetch the results by passing a third parameter which preg_match() will fill as an array.

Named Capture Groups

```
# Extract the number from a date string
```

```
preg_match("/(?P<year>\d{4})-(?P<month>\d{2})-(?P<day>\d{2})/", "2012-10-20", $resul
```

Now the $result array will additionally to the position matches 1, 2 and 3 contain the keys "year", "month" and "day". The advantage is never having to think of the capture positions anymore when you modify the expression!

4. Check Processing Errors!

While it might often be unimportant be aware that applying a regular expression might fail due to PCRE constraints. This usually happens when matching overly long strings or strings with faulty encoding. The only way to notice that preg_match() was not able to check the string is by calling

```
preg_last_error()
```

Only if it returns PREG_NO_ERROR you got a safe result! Consider this when using preg_match() for security purposes.